Masterarbeit zur Erlangung des akademischen Grades

„Master of Arts (M.A.)“

im Studiengang Medienwissenschaft

Berechenbarkeit als Sphäre digitaler Medien

Computability as the Sphere of Digital Media

Thomas Nückel
Höglwörtherstr. 369
81379 München
thomas.nueckel@hu-berlin.de
Matrikelnummer: 553745
Wintersemester 2016/17

1. Gutachter: Prof. Dr. Wolfgang Ernst
2. Gutachter: Dr. Jan Claas van Treeck

Hinweise zur Arbeit

Die Seitenzahlen in dieser Arbeit wurden aus Gründen der Handhabbarkeit entsprechend den Seitenzahlen im PDF-Dokument gesetzt.

Hervorhebungen in Zitaten sind, wenn nicht in eckigen Klammern am Ende des Zitats besonders vermerkt, aus dem Original übernommen.

Das Literaturverzeichnis gibt jeweils die Jahreszahl der Erstveröffentlichung (Vorträge mitberücksichtigt) an. Die Jahreszahl der verwendeten Auflage bzw. Übersetzung findet sich, wenn vorhanden, nach dem Titel des Werks.

Zitate aus Onlinequellen wurden durch diesbezügliche Vermerke in Klammern besonders gekennzeichnet.

Danksagung

Der Verfasser dankt für fachlichen Rat, Korrekturen und zahlreiche Hinweise Prof. Dr. Karl-Georg Niebergall von der Humboldt-Universität zu Berlin, Prof. Dr. Wilfried Sieg von der Carnegie Mellon University, Prof. Dr. Horst Völz, Prof. Dr. Siegfried Zielinski von der Staatlichen Hochschule für Gestaltung Karlsruhe, Dr. Nikita Braguinski, Dr. Marcus Burkhardt, Dr. Stefan Höltgen, Dr. Tiago da Costa e Silva, Dr. Robert Smid, Christoph Borbach, Manuel Günther, Moritz Hiller, Pascal Lünnemann, Johannes Maibaum, Milena Nikolova, Eva-Maria Raffetseder, Matthias Wannhoff und Marco Weßnigk, sowie Jacqueline Franke für ihre organisatorische Unterstützung.

Inhaltsverzeichnis

Das Niveau einer Wissenschaft bestimmt sich daraus, wie weit sie einer Krisis ihrer Grundbegriffe *fähig* ist.[1]
Martin Heidegger

1. Die Digitalisierung und ihre Grenzen

Digitale Geräte finden sich nahezu überall, im Privatleben wie in Arbeitswelt und Öffentlichkeit. Entsprechend sind digitale Rechner auch aus dem wissenschaftlichen Betrieb nicht mehr wegzudenken. Dies beginnt beim Verfassen von Textarbeiten und endet bei Disziplinen wie synthetischer Biologie, Gentechnik, Nanotechnologie oder der modernen Teilchenphysik, die es allesamt ohne Rechnerunterstützung in dieser Form gar nicht geben könnte. Überhaupt gilt, dass die Verwaltung der Universitäten, der Gemeinden und Städte, der kleinen und großen Gesellschaften ohne die digitalen Geräte nicht länger möglich wäre – zumindest nicht ohne erhebliche soziale Veränderungen. Die Computer werden dabei immer uneinsehbarer, verschwinden hinter den von ihnen simulierten Zeichen, Graphiken, Bewegtbildern und Klängen, machen im Gebrauch ihre eigene Technik unsichtbar. Phänomene wie *Deep Learning* und selbstlernende neuronale Netze, das *Semantic Web* mit seinen Ontologien, das entstehende Internet der Dinge zeugen von der doppelten Tendenz der digitalen Maschinen, sich zugleich dem menschlichen Zugriff zu entziehen und sich der Lebenswelt als formierende Kraft überzustülpen. Angesichts der tausend Verkleidungen, in denen Rechner auftreten können, greifen nutzungsorientierte Ansätze stets zu kurz, wenn es um ein Verständnis der hochtechnischen Medien selbst gehen soll. Dieses Absehen vom Technischen geht dabei so weit, dass an ein Jenseits des Digitalen kaum noch gedacht werden kann. „Die Technikgeschichte auf ihrem Triumphzug produziert mithin selber die Illusion, daß es andere Prinzipien als die in Digitalcomputern realisierten gar nicht geben könne.“[2]

Um dem entgegenzuwirken, nimmt diese Arbeit den Begriff des Digitalen von zwei Seiten in die Zange: Für den Zugang von innen bietet sich die Turingmaschine von 1936 in besonderer Weise an, ist sie doch Instrument und Ergebnis einer „Untersuchung der theoretischen Möglichkeiten und Grenzen digitaler Rechenmaschinen.“[3] Gleichzeitig wird diese Analyse auch von außen geführt, d. h. vom Bereich dessen aus, was mit digitalen Maschinen *nicht* möglich ist, sondern sich als das Unberechenbare bislang dem medialen Zugriff entzieht. Für den Blick von außen werden u. a. Modelle aus dem Bereich der Hypercomputation

[1] Heidegger 1927, 9.
[2] Kittler 1996a, 123.
[3] Turing 1947, 186.

vorgestellt, die in der Lage sein sollen, auch für digitale Maschinen Unberechenbares zu berechnen. Ziel dieses doppelten Zugangs ist die Ausarbeitung und medientheoretische Analyse einer Sphäre der Berechenbarkeit, die all das umschließt, was mit digitalen Maschinen überhaupt möglich ist.

Um diese Sphäre digitaler Medien zu entwickeln, müssen zunächst zumindest einige Eigenschaften der Turingmaschine im Detail betrachtet werden (Kapitel 2). Anhand von Gödels Lesart der Turingmaschine als mechanische Prozedur wird daraufhin der Bezug dieses Konzepts zu realen Maschinen herausgearbeitet (Kapitel 3). Der Schwerpunkt der Arbeit besteht dann aus der Diskussion der logischen Möglichkeit von Hypercomputation, der Einführung in verschiedene konzeptuelle Hypercomputer und der Darlegung der Erkenntnisse, die sich anhand von Hypercomputation zur Turingmaschine gewinnen lassen (Kapitel 4). Auf dieser Grundlage wird das Konzept der Sphäre der Berechenbarkeit ausformuliert. Dies geschieht anhand einer kritischen Beleuchtung der Frage, ob Maschine und Medium im Kontext des hier verhandelten Ansatzes der Sphäre notwendig zusammenfallen müssen oder begrifflich getrennt behandelt werden können (Kapitel 5). Da sich zeigen wird, dass der Entwurf des Konzepts der Sphäre keine Entscheidungen über die Eigenschaften und das Verhältnis von Medien und Maschinen notwendig macht, soll dieses offene Sphärenkonzept medientheoretisch geerdet werden, indem einige Implikationen und Gedanken, die mit diesem einhergehen, diskutiert werden (Kapitel 6). Im letzten Teil dieser Arbeit wird die so als Sphäre entwickelte Grenze der Operativität von Medien kurz und abschließend zusammengefasst (Kapitel 7).

Um aber anhand der Sphäre der Berechenbarkeit Erkenntnisse über digitale Medien gewinnen zu können, muss der Begriff des Digitalen seine scheinbare Selbstverständlichkeit verlieren – denn, wie schon Platon es seinem Lehrer und Protagonisten Sokrates in den Mund legte, suchen wir nur, was wir nicht zu wissen glauben.[4] In diesem Sinne soll der Begriff des Digitalen durch diesen alternativen Zugang neu beleuchtet und zugleich erst wieder fragwürdig gemacht werden. Damit kann diese Arbeit als Versuch verstanden werden, einem Gedanken Kittlers Rechnung zu tragen, der da schrieb: „Computer werden so tun, als ob sie riechen, duften, singen oder sonstwas. Sie werden nie so tun, als ob sie rechnen. Aber genau das würde ich wünschen: daß sie ihr eigenes Rechnen zugänglicher machen.“[5]

[4] Vgl. Platon, 109e.
[5] Kittler 1992, 88.

2. Die Turingmaschine

Alan Mathison Turings Turingmaschine von 1936 gilt gemeinhin als theoretisches Konstrukt, als eine rein konzeptuelle Papiermaschine. Tatsache ist, dass Turing diese Maschine in ihrer ursprünglichen Form auf dem Papier und ohne jede Art von tatsächlichem Schaltplan, d. h. rein auf Basis von Symbolen und deren Verknüpfung entwickelte. Entsprechend behandelt die anwendungsorientierte Informatik diesen theoretischen Entwurf Turings recht stiefmütterlich. Nichtsdestotrotz ist der diesbezügliche Text Turings – ‚On Computable Numbers, with an Application to the Entscheidungsproblem'[6] – nicht nur eines der Gründungsdokumente der die Materialität von Medien betonenden Medienwissenschaft, sondern auch für die moderne theoretische Computerwissenschaft der *fons et origo*:[7]

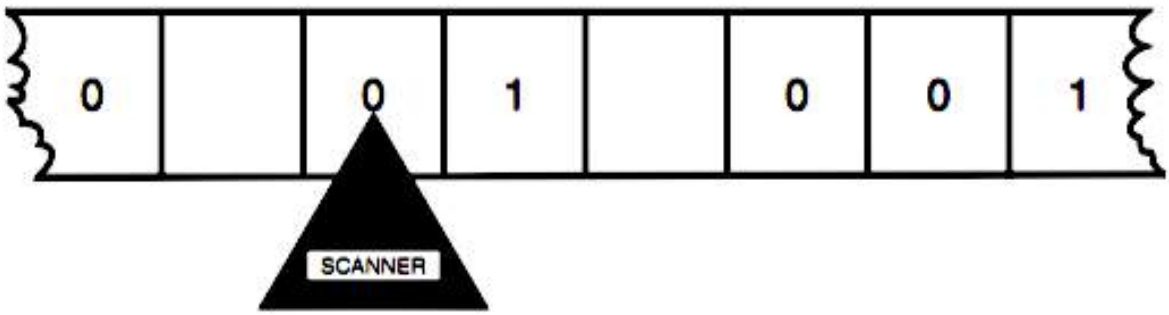
> 'On Computable Numbers' is regarded as the founding publication of the modern science of computing. It contributed vital ideas to the development, in the 1940s, of the electronic stored-programme digital computer. […] In this one article, Turing ushered in both the modern computer and the mathematical study of the uncomputable.[8]

Die Arbeit Turings öffnete laut Copeland also das Spannungsfeld zwischen den tatsächlichen modernen Computern und dem mathematischen Studium dessen, was *uncomputable* ist, d. h.: des Unberechenbaren. Es ist dieses Spannungsfeld, in dem die folgende Untersuchung stattfindet und es ist die von Alan Turing in die Welt gesetzte Turingmaschine, von der sie ihren Ausgang nimmt. Es wird daher zunächst um Aufbau und Funktionsweise dieser Maschine gehen (Kapitel 2.1), bevor zwei Funktionen dargestellt werden, die von der Turingmaschine nicht berechnet werden können. Dabei handelt es sich um das von Turing entwickelte *Satisfactoriness Problem* (Kapitel 2.2.1) und das in diesem Kontext geradezu klassische Halteproblem (Kapitel 2.2.2). Als dritter Punkt wird ein kurzer Übergang zu Gödels Lesart der Turingmaschine gegeben (Kapitel 2.3). In dem auf diesen Abschnitt folgenden dritten Kapitel werden dann die Unterschiede zwischen der Auffassung Gödels und der Turings stark gemacht. Hierzu muss vorausgeschickt werden, dass die dabei vorgestellte Sichtweise Turings – die an einigen Punkten als Gegenpol zur Auffassung Gödels dienen wird – eine durch seine Exegeten vermittelte ist. Es wird sich zeigen, dass das in diesem Kapitel ausgehend von der Analyse Turings Beschriebene sich weniger von den Auslegungen Gödels unterscheidet, als es aus Sicht mancher Exegeten scheinen mag.

[6] Vgl. Turing 1936.
[7] Vgl. Copeland/Shagrir 2015, 1.
[8] Copeland 2004, 6.

2.1 Aufbau und Funktionsweise

Die Maschine, die Turing in ‚On Computable Numbers' entwirft, ist, was ihre elementaren Bauteile betrifft, verblüffend einfach. Sie besteht, in ihrer abstrakten Form, aus nur zwei Teilen: Einem Scanner und einem endlosen Band. Strenggenommen zählt für Turing auch das Band nicht zur Maschine, sondern die Maschine wird mit diesem Band versorgt: „The machine is supplied with a "tape" (the analogue of paper) running through it […].“[9]



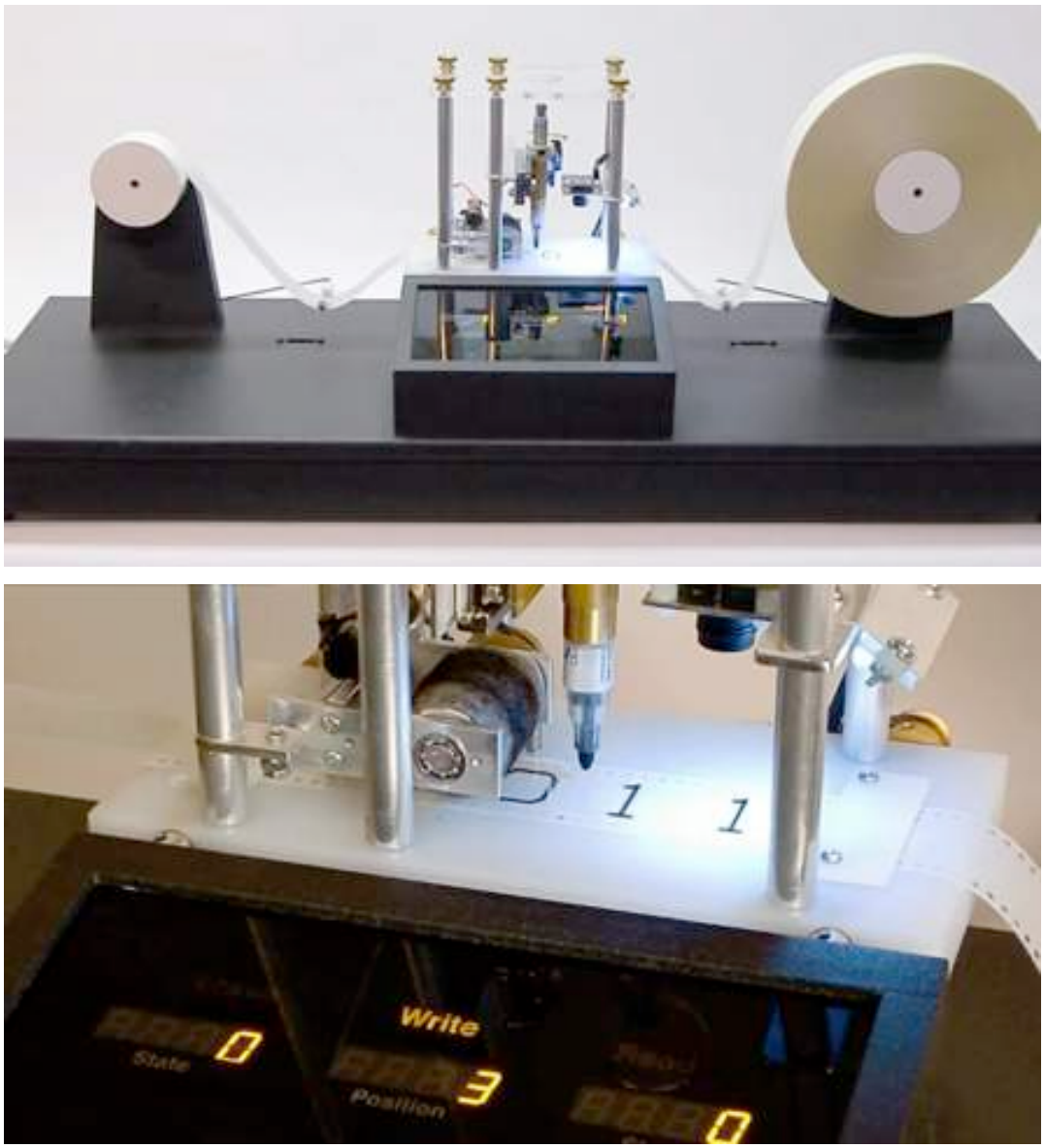
Schematische Darstellung der Turingmaschine mit Scanner und Band.[10]

Wie in der Abbildung dargestellt, ist das Band in Kästchen unterteilt. Es kann relativ zum Scanner nach links und rechts bewegt werden, so dass stets genau ein Kästchen unter dem Scanner zum Liegen kommt. In Turings Konzept wird das Band bzw. der Scanner bei jedem Schritt nie weiter als nur ein Kästchen bewegt, denn mehr wird für die Umsetzung der universalen Turingmaschine am Ende nicht nötig sein. Jedes Kästchen kann ein einzelnes Symbol enthalten, z. B. ‚0' oder ‚1', aber auch jedes andere arbiträre Symbol aus einem endlichen Alphabet.[11] Für Turing gilt das leere Kästchen nicht als Symbol: „In some of the configurations in which the scanned square is blank (*i.e.* bears no symbol) […].“[12] Diese Unterscheidung ist allerdings rein methodischer Natur, so dass moderne Schreibweisen auch das leere Kästchen als ein gewöhnliches Symbol auffassen. Das zweite Bauteil, der Scanner, dient dazu, die Symbole auf dem Band zu lesen, zu schreiben und auszuradieren. Etwas weniger anthropozentrisch formuliert heißt das: Der Scanner scannt, druckt und löscht die Symbole auf dem Band und bearbeitet dabei pro Operationsschritt immer nur ein Kästchen.

---

[9] Turing 1936, 59.
[10] Abbildung aus Copeland 2004, 7.
[11] Turing spezifiziert das in seiner Arbeit verwendete Alphabet natürlich auf eine bestimmte Weise. Das ändert aber nichts daran, dass jedes dieser Zeichen auch durch ein willkürliches anderes zu ersetzen wäre. Solange die Beziehungen zwischen den Zeichen nicht verändert werden, kommt es dabei zu keinem Unterschied im Verhalten.
[12] Turing 1936, 59.

Diese Bilder einer realen und voll operationsfähigen Turingmaschine verdeutlichen den beschriebenen Aufbau:



Reale Umsetzung der Turingmaschine.[13]

Im oberen Bild zu sehen ist die gesamte Maschine mit zwei Trommeln links und rechts für das Band, hier von etwas über 300 Metern Länge. Im unteren Bild von links nach rechts die Radiertrommel zum Löschen, ein Schreibstift zum Drucken und die Kamera zum Scannen des Bandes. Da hier zwei Schrittmotoren das Band bewegen und jeweils eine volle Umdrehung dieser Motoren den Abstand vom Mittelpunkt eines Symbols zum nächsten definiert, hat der Erbauer des abgebildeten Apparats auf eine Markierung für die einzelnen Kästchen verzichtet, wohl auch damit der Scanner die Trennstriche zwischen den Kästchen nicht mit den Einsen auf dem Band verwechselt.

[13] Abbildungen von Guizzo 2010 (Internetquelle).

Wie aber wird festgelegt, zu welchem Kästchen eine Turingmaschine als nächstes wechseln soll und wie sie mit dem Symbol dort zu verfahren hat? Woher erhält die Maschine ihre Instruktionen?

Zusätzlich zu den beschriebenen Operationen des Scannens, Druckens und Löschens ist der Scanner in der Lage, seinen *Zustand* zu speichern und nach festen Regeln zu verändern.[14] Wie Turing zeigt, lassen sich Tabellen bilden, die das Verhalten der Maschine für jeden möglichen Fall eindeutig festlegen:

Table 1

State	Scanned Square	Operations	Next State
a	blank	P[0], R	b
b	blank	R	c
c	blank	P[1], R	d
d	blank	R	a

Tabelle einer Turingmaschine.[15]

Die Maschine kann zwar stets nur ein einzelnes Kästchen scannen: „The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its *m*-configuration [*State* in der Tabelle] the machine can effectively remember some of the symbols which it has "seen" (scanned) before.“[16] Das heißt, der Zustand (in der obigen Abbildung ‚State' genannt) ist eine Zusammenfassung der bisher geleisteten Operationsschritte zumindest insofern, als dass alle für die Bearbeitung des nächsten Symbols nötigen Informationen darin komprimiert sind – eine Art Gedächtnis also, wie die eben zitierte Beschreibung Turings selbst nahelegt.

Die obige Tabelle kann in zwei Hälften geteilt werden: In den beiden linken Spalten findet sich der Ist-Zustand der Maschine, in den zwei rechten Spalten die Anweisungen, die sie ausführen soll. Um zu begreifen, wie die Ausführung einer solchen Tabelle vonstatten zu gehen hat und um die dabei wesentlichen Zusammenhänge nachzuvollziehen, ist es ratsam, Instruktionen wie die hier angegebenen einmal von Hand auszuführen.

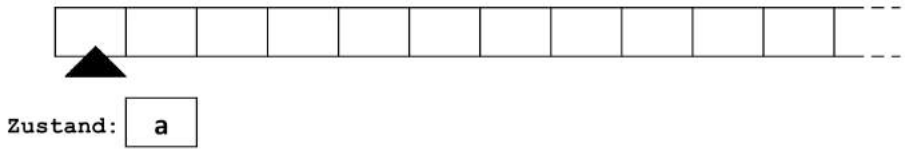
---

[14] Tatsächlich definiert Turing in ‚On Computable Numbers' eine Hierarchie aus drei verschiedenen Zuständen der Maschine, die er als ‚configurations' bezeichnet. Der Einfachheit halber wird in diesem Kapitel ausschließlich der Terminus ‚Zustand' verwendet, was sich durchwegs auf das bezieht, was Turing ‚*m*-configuration' nennt, was in der Übersetzung Bernhard Siegerts entsprechend ‚*m*-Zustand' heißt und im modernen Turingmaschinen-Jargon als ‚State' bezeichnet wird. Die Übersetzung Siegerts findet sich in Dotzler/Kittler 1987, 17-60.

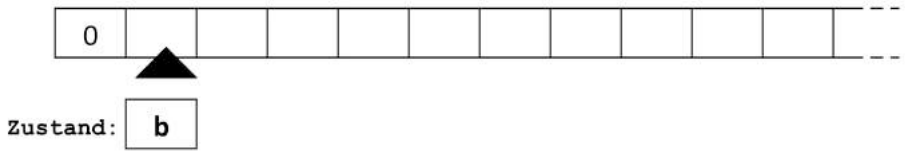
[15] Abbildung aus Copeland 2004, 8. Die Tabelle entspricht funktional der ersten von Turing behandelten (vgl. Turing 1936, 61). Copeland verwendet aus Gründen der Lesbarkeit eine modernere Nomenklatur.

[16] Turing 1936, 59.

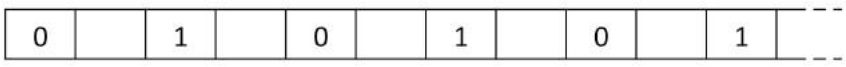
Zu Beginn ist die Turingmaschine, oder der Mensch, der vorhat, den Anweisungen der Tabelle zu folgen, im Zustand *a* und das Kästchen unter dem Scanner ist leer.[17]



Nun wird in den beiden linken Spalten der Tabelle nach dieser Kombination aus Zustand und gescanntem Symbol gesucht. Wie sich zeigt, enthält gleich die erste Zeile die Kombination aus Zustand *a* und dem leeren Kästchen ‚blank'. Daher müssen die dazugehörigen Anweisungen ausgeführt werden. So steht in der dritten Spalte: ‚*P*[0], *R*'. Das heißt: ‚Drucke (Print) das Symbol ‚0' auf das Band und rücke den Scanner ein Kästchen nach rechts'. Die vierte Spalte weist zudem an, in einen neuen Zustand zu wechseln, nämlich in den Zustand *b*.



Da die Maschine nun zwar erneut ein leeres Kästchen scannt, sich aber jetzt im Zustand *b* befindet, führt sie im nächsten Schritt nicht mehr die Instruktionen aus der ersten Zeile der Tabelle aus. Stattdessen findet sie die nun vorliegende Kombination aus Zustand und Symbol – *b* und ‚blank' – in der zweiten Zeile. Entsprechend führt die Maschine jetzt die in der zweiten Zeile angegebenen Instruktionen aus und rückt, ohne zu drucken, ein Kästchen nach rechts. So fortfahrend kann der Rechner, ob Maschine oder Mensch, stets die vorliegende Kombination aus Zustand und gescanntem Symbol in der Tabelle suchen und findet dort die nächsten Anweisungen. Damit ist jeder Schritt vollständig von den Regeln der Tabelle bestimmt. Führt der Rechner die Anweisungen korrekt aus, füllt sich das Band mit einer Serie von alternierenden Nullen und Einsen, zwischen denen jeweils ein Kästchen frei bleibt.



Wie bereits erwähnt, lässt sich das dahinterliegende Prinzip durch den eigenhändigen Nachvollzug weit besser begreifen als durch theoretisierendes und distanziertes Betrachten.

[17] Dieser Anfangszustand wird standardmäßig zusätzlich zur Tabelle angegeben. Der hier gewählte entspricht dem Zustand, in dem auch Turing die Abarbeitung der Tabelle in 1936 beginnen lässt.

Da es in der vorliegenden Untersuchung um dieses Prinzip geht und um das, was mit diesem Prinzip *nicht* erzeugt werden kann, wird keine weitergehende Beschreibung der elementaren Funktionen der Turingmaschine gegeben. Verwiesen sei stattdessen auf Turings Arbeit selbst, auf die gelungene Einführung ‚Computable Numbers: A Guide' von Copeland und auf Weizenbaums klare Beschreibung, dem zu diesem Zweck ein paar Steine und eine Rolle Klopapier ausreichen.[18] Die zwei Aspekte des der Turingmaschine zugrundeliegenden Prinzips lassen sich denn auch schon anhand dieser kurzen Einführung aufzeigen:

Ein Mensch, der die obige Tabelle ausführt, wird schnell feststellen, dass es keine Bedingung gibt, unter der die Arbeit zu einem Halt kommt. Stattdessen würde sich, sofern keine äußeren Umstände die Abarbeitung unterbrechen, die Ausführung der Tabelle unendlich weiter fortsetzen. Das bedeutet: Mittels einer *endlichen* Tabelle – was eine der Grundanforderungen an jeden Algorithmus darstellt – lassen sich Vorschriften auch für die Berechnung *unendlicher* Ausgaben [sic] vollständig angeben. Aus diesem Grund ist die Turingmaschine konzeptuell in der Lage, nicht nur eine unendliche Folge alternierender Nullen und Einsen anzuschreiben, sondern auch die Ausdrücke reeller Zahlen wie π oder *e*.[19] Dieser Umstand wird gemeinhin angeführt, um den Unterschied zwischen reinen Papiermaschinen wie der Turingmaschine und realen Maschinen wie den heute ubiquitären digitalen Computern zu ziehen: Während die Turingmaschine in einer logischen Zeit abläuft und daher als Konzept durchaus in der Lage ist, unendliche Ausdrücke anzuschreiben, sind es ihre realen Umsetzungen nicht, da sie einer entropischen Zeit unterliegen und ihre endlichen Ressourcen nach endlich vielen Schritten aufgebraucht haben werden (vgl. auch Kapitel 4.3.1).

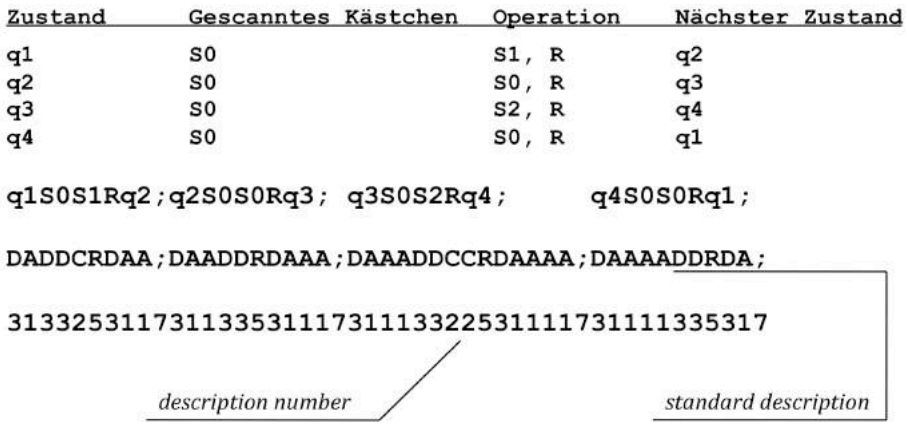
Die Ausführung einer solchen Tabelle zeigt außerdem in aller Klarheit den *mechanischen* Charakter dieser Tätigkeit. Jeder Arbeitsschritt ist vollständig durch den momentanen Zustand und das augenblicklich gescannte Symbol bestimmt. Es ist kein Wissen um die Bedeutung oder irgendeine Interpretation der Arbeitsschritte und ihres Ergebnisses notwendig, um zum richtigen Ergebnis zu kommen. Es genügt, die Tabelle ohne Nachdenken abzuarbeiten. In den Worten Lacans: „[…] [W]enn die Maschine nicht denkt, dann ist es klar, daß wir selbst auch nicht denken in dem Moment, in dem wir eine Operation ausführen. Wir folgen exakt denselben Mechanismen wie die Maschine."[20]

Dies sind die zwei Kernaspekte der von Turing entwickelten Analyse. Folgerichtig fragt Turings Arbeit danach, welche *unendlichen* Zahlausdrücke sich auf *mechanische* Weise anschreiben lassen – und welche nicht.

---

[18] Vgl. Turing 1936, 59-66; Copeland 2004, 5-57 und Weizenbaum 1977, 80-88.
[19] Vgl. Turing 1936, 82.
[20] Lacan 1955, 385.

Wie Turing in seiner Arbeit zeigt, lassen sich Tabellen der bereits beschriebenen Art nicht nur erstellen, sondern in sogenannte *standard descriptions* und *description numbers* umwandeln:



Umwandlung einer Tabelle zur *standard description* und zur *description number*.

Die Umwandlung geschieht wie abgebildet in drei Schritten und besteht ausschließlich aus dem Erstellen zueinander isomorpher Zeichenketten. Wie zu sehen ist, wurden die Symbole im Vergleich zur ursprünglichen Tabelle ausgetauscht: Statt *a*, *b*, *c* und *d* werden nun q_1 bis q_4 zur Bezeichnung der Zustände verwendet. Außerdem wird das leere Kästchen nicht mehr durch ‚blank', sondern durch ‚*S*0' – für Symbol 0 – gekennzeichnet. Entsprechend ist die ‚0' nun durch ‚*S*1' und die ‚1' durch ‚*S*2' ersetzt worden. Wie bereits betont, führen solche Substitutionen auf der Ebene der Zeichen zu keinerlei Veränderung im Verhalten der Maschine, solange die Relationen zwischen den Zeichen beibehalten werden.

Der erste Schritt der Umwandlung besteht nun darin, die Zeichen in jeder der einzelnen Zeilen gleichsam aus den Spalten zu lösen und sie direkt nebeneinander zu gruppieren. Damit auch nach dieser Konkatenation nachvollziehbar ist, wo eine Zeile endete und die nächste begann, wird jeweils ein Semikolon als Markierung gesetzt. Turing beschreibt im Detail, wie die nun eindimensionale Instruktionsanweisung weiter in Buchstaben – die *standard description* – und Zahlen – die *description number* – zu übersetzen ist.[21] Für hier ist nur wesentlich, dass sich der Gehalt der Instruktionen bei keinem der Übersetzungsschritte ändert. Auch die *standard description* und die *description number* können ganz wie die Tabelle ausgeführt werden und erzeugen dabei exakt dasselbe Ergebnis. In diesem Sinne ist die obige Umwandlung der Tabelle nichts anderes als das *Kompilieren* von einer für Menschen verständlicheren Sprache in eine abstraktere und maschinengerechtere.[22]

[21] Vgl. Turing 1936, 66-68.
[22] Vgl. Copeland 2004, 12 und Mayer-Lindenberg 1998, 170f.

Damit ist schon angeklungen, wozu die Umwandlung der Tabelle in die *standard description* bzw. die *description number* dient:

Sollte die obige Tabelle in einer realen Turingmaschine umgesetzt werden, müsste das Programm, das durch diese Tabelle angegeben wird, lediglich mit logischen Gattern und Speicherzellen im Kopf des Scanners fest verdrahtet werden.[23] Die so gebaute spezielle Turingmaschine könnte, wenn der Scanner außerdem über die gleichen Lese-, Schreib- und Transportgeräte verfügte wie die oben bebilderte reale Umsetzung einer Turingmaschine, das Programm der Tabelle ausführen und würde dann mit jeweils einem Zwischenraum alternierend Nullen und Einsen auf das Band schreiben.

Was aber, wenn statt dieses Programms eine andere Tabelle von der Maschine ausgeführt werden sollte, wenn stattdessen zum Beispiel π oder *e* berechnet werden soll? Auf symbolischer Ebene wären mit einem solchen Austausch keine großen Schwierigkeiten verbunden. Es hätte gereicht, die Symbole der ursprünglichen Tabelle durch die einer entsprechenden anderen Tabelle, mittels derer π oder *e* berechnet werden können, zu ersetzen. Im Falle der Umsetzung als realer Maschine allerdings läge das Programm eben mehr oder weniger fest verdrahtet im Kopf des Scanners, so dass entweder jeweils mühevoll die Drähte umgesteckt oder ähnlich einer Schreibmaschine mit Kugelkopf verschiedene Wechselköpfe mit Programmen bereit gehalten werden müssten, und zwar für jede mögliche Turingmaschine ein anderer Kopf – was eine abzählbar unendlich große Menge an Wechselköpfen bedeuten würde. Beide Lösungen erscheinen nicht nur unpraktisch, sie führen auch in Bezug auf die Möglichkeiten der Turingmaschine nicht weiter. Wenn eine spezielle Turingmaschine eine bestimmte Zahl nicht berechnen kann, dann liegt das doch schlicht daran, dass sie dafür das falsche Programm, also den falschen Wechselkopf hat. Wer sagt, dass nicht eine andere spezielle Turingmaschine für die Lösung des gewünschten Problems gebaut werden kann oder im unendlich großen Archiv der Wechselköpfe schon bereit liegt?

Turings Leistung und sein Schritt von der speziellen zur universalen Turingmaschine bestehen nun darin, das Problem zurück in das Symbolische zu ziehen und es so zu lösen. Wie er zeigt, können sowohl die *standard description* als auch die *description number* – d. h. die Tabelle – statt als Hardwarelösung fest verdrahtet zu werden, selbst als Symbole auf das Band einer Turingmaschine geschrieben werden. Das hat allerdings nicht zur Folge, dass es gar kein fest verdrahtetes Programm mehr gäbe! Für die Umsetzung einer solchen Tabelle mit bedingten Sprüngen zwischen verschiedenen Zeilen reichen Speicher- und Übertragungselemente wie das Band und die Lese- und Schreibköpfe des Scanners allein

[23] Ein Taktgeber zur Synchronisation wäre bei den allermeisten realen Umsetzungen ebenso notwendig.

schon nach Kittlers Mediendefinition nicht aus.[24] Was aber entfällt, ist die Notwendigkeit, beim Wechsel des Programms auch die Hardware zu wechseln. Wie Turing beweisen kann, reicht ein einziges spezielles fest verdrahtetes Programm im Scanner aus, um jede mögliche Turingmaschine, deren Beschreibungsnummer (also die *standard description* oder die *description number*) auf das Band geschrieben wurde, zu lesen und auszuführen. Von diesem Moment an ist es nicht mehr nötig, Verbindungen und Kabel umzustecken, wenn ein anderes Programm gewünscht wird. Es genügt, die Symbole auf dem Band auszutauschen. Kurz: Eine einzige aber universale Turingmaschine reicht aus, um jede spezielle Turingmaschine zu imitieren. Damit ist der Schritt von *programme-controlled*, wie die fest verdrahteten frühen Rechner Colossus (1943) und ENIAC (1945) es noch sein sollten, zu *stored-programme*, also zur variablen Speicherprogrammierung im modernen Computer, vollzogen.[25]

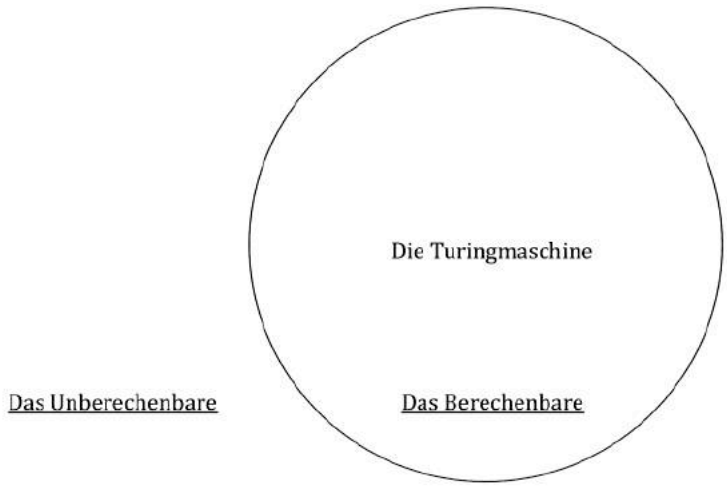
Mit einem einzigen und festen Programm kann die *universale Turingmaschine* alles berechnen, was spezielle Turingmaschinen berechnen können.[26] Nach der Church-Turing These heißt das: Die universale Turingmaschine kann all das berechnen, was ein menschlicher Mathematiker berechnen kann.[27] Es existieren verschieden starke Auslegungen dieser These, und auch Turings Meinung diesbezüglich wandelte sich mit der Zeit. Als Fakt lässt sich aber festhalten, dass sich bis heute keine Rechnungen haben angeben lassen, die ein Mensch hätte lösen können, für die sich aber keine Turingmaschine finden ließe. Somit gilt zumindest empirisch, dass die vom Begriff der Turingmaschine und vom Begriff der Berechenbarkeit abgesteckten Bereiche gleiche Extension aufweisen.

Die Turingmaschine als Definition des Berechenbaren.

[24] Vgl. u. a. Kittler 1986, 352.
[25] Vgl. Copeland 2004, 8.
[26] Vgl. ebd., 15.
[27] Vgl. dazu die äußerst gelungene Zusammenfassung zur Church-Turing These mit Auflistung auch der verbreiteten Irrtümer über ihre Tragweite bei Ord 2002, 10-14.

2.2 Unberechenbare Zahlen

Folgt man der Church-Turing These, dann war es die Leistung Turings, eine wohldefinierte Beschreibung dessen zu liefern, was überhaupt effektiv berechenbar ist. In den Worten Kittlers: „Diese Mathematik vollständig beschreibbarer endlicher Schritte auf einem Papierband, das lediglich das Lesen, Schreiben und Löschen einzelner Zeichen erlaubt, erwies sich als Inbegriff von Berechenbarkeit überhaupt.“[28] Soll die Definition von Berechenbarkeit aber nicht trivial bleiben, so muss es notwendig auch ein Außerhalb des von dieser Definition abgesteckten Bereiches geben, kurz: Es muss auch unberechenbare Funktionen geben.
Wie bereits erläutert wurde, gelten auch Zahlen mit unendlichem Dezimalausdruck als mittels der Turingmaschine berechenbar. Die Menge dieser berechenbaren reellen Zahlen, unter die eben auch π und *e* fallen, ist *abzählbar* unendlich. Dieser Umstand lässt sich leicht beweisen: Wie gezeigt, kann jeder Turingmaschine genau eine *standard description* bzw. *description number* zugewiesen werden. Da die *description number* jeweils eine natürliche Zahl ist, gibt es nur abzählbar unendlich viele *description numbers*. Somit kann es auch nur abzählbar unendlich viele Turingmaschinen geben. Da zudem jede Turingmaschine nur eine einzige berechenbare Zahl ausgibt, kann es auch nur abzählbar unendlich viele berechenbare Zahlen geben. Dem gegenüber steht eine *überabzählbar* unendliche Menge an unberechenbaren Zahlen.[29] Das heißt, obwohl der Bereich des Berechenbaren aus einer unendlichen Menge von Zahlen besteht, übertrifft die Menge der unberechenbaren Zahlen erstere in gleicher Weise wie die Menge der reellen Zahlen die Menge der rationalen bzw. natürlichen.



Der Bereich des Berechenbaren als eine Insel im Unberechenbaren.

[28] Kittler 1992, 263.
[29] Vgl. Copeland 2004, 35. Tatsächlich produziert nach Turings Definition sogar nur jede zirkelfreie Turingmaschine eine berechenbare Zahl (vgl. besonders Kapitel 2.2.1.2 und 2.2.1.5). Dies ändert nichts an der Stichhaltigkeit der Argumentation.

2.2.1 Das Satisfactoriness Problem

Wie aber sind solche *unberechenbaren Zahlen* vorzustellen? Handelt es sich dabei nicht um ein Oxymoron? Um diese Fragen zu klären, wird als erstes Beispiel für eine solche unberechenbare Funktion bzw. Zahl das *Satisfactoriness Problem* vorgestellt. Dieses stammt von Turing selbst und wird in ‚On Computable Numbers' zentral verhandelt.[30] Für das Verständnis des *Satisfactoriness Problems* ist es notwendig, zunächst zwei recht trockene Eigenheiten vorzustellen, die für Turings Entwurf und seine Argumentation maßgeblich sind, auch wenn sie so in modernen Varianten der Turingmaschine nicht mehr verwendet werden. Das ist zum einen die Unterscheidung zwischen Symbolen erster und zweiter Art, zum anderen die Trennung zwischen zirkulären und zirkelfreien Turingmaschinen.

2.2.1.1 Symbole erster und zweiter Art

In seiner Konzeption unterscheidet Turing zwischen Symbolen der ersten und Symbolen der zweiten Art. Unter die *Symbole der ersten Art* fallen lediglich die Ziffern ‚0' und ‚1' der Binärzahl, die von der Turingmaschine als Ergebnis ausgegeben wird. Diese Symbole werden in Turings ursprünglichem Konzept nur in jedes zweite Kästchen auf dem Band geschrieben, ganz wie in der Tabelle aus dem vorhergehenden Kapitel. Da diese Binärziffern bei Turing nie mehr gelöscht werden, nennt er die für sie vorgesehenen Felder[31] *F-Felder*.[32] Zwischen den *F-Feldern* liegen die sogenannten *E-Felder*. Diese dienen der Maschine lediglich für Markierungen oder zum Festhalten von Zwischenergebnissen, sind aber niemals Teil der letztendlich berechneten Zahl. Auf die *E-Felder* werden entsprechend nicht ‚0' und ‚1', sondern Symbole wie ‚2', ‚*', ‚*x*', usw. und das leere Kästchen geschrieben. Bei diesen für die *E-Felder* vorgesehenen Symbolen handelt es sich um die *Symbole der zweiten Art*.[33] Diese terminologische Unterscheidung hat sich in modernen Arbeiten nicht durchgehalten. Dennoch ist sie für Turings Argumentation entscheidend. So baut auf der Distinktion zwischen Symbolen der ersten Art und der zweiten Art schon Turings Definition der zirkulären und der zirkelfreien Maschinen in maßgeblicher Weise auf.

---

[30] Vgl. Turing 1936, 72f.
[31] Mit Feld ist nichts anderes gemeint als ein Kästchen auf dem Band der Turingmaschine. Feld und Kästchen sind in diesem Kontext synonym.
[32] Wobei sich *F-Felder* vom Wort *Fixed* herleiten und die noch folgenden *E-Felder* von *Erasable*.
[33] Vgl. ebd., 60 und 63. Vgl. auch Copeland 2004, 32f.

2.2.1.2 Zirkuläre und zirkelfreie Turingmaschinen

Die zweite Unterscheidung macht Turing zwischen zirkulären (*circular*) und zirkelfreien (*circle-free*) Turingmaschinen. Wie Copeland bestätigt, ist auch diese Terminologie Turings etwas verwirrend und kam in den Arbeiten anderer nicht mehr zur Anwendung.[34]

Im Prinzip gilt die einfache Unterscheidung: *Zirkuläre* Maschinen drucken jeweils nur eine endliche Anzahl von Symbolen der ersten Art. *Zirkelfreie* Maschinen dagegen drucken stets eine unendliche Anzahl von Symbolen der ersten Art.

Genau dies aber mag auf den ersten Blick verwirren: Sollten nicht zirkuläre Maschinen unendlich viele Symbole drucken, ganz so wie die durch obige Tabelle verkörperte Maschine, die immer wieder zirkulär ihre Tabelle bis ins Unendliche abarbeitet? Wäre nicht ebenso eine zirkelfreie Maschine eine, die nach einer endlichen Anzahl von Arbeitsschritten stehenbleibt, da sie sich eben nicht in einen Zirkel begibt? Turings Definition ist aber tatsächlich wie oben stehend genau andersherum.

Die Erklärung hierfür ergibt sich aus der Einteilung in Symbole erster und zweiter Art. *Zirkulär* nämlich ist eine Maschine, die sich beim Schreiben von Symbolen der zweiten Art in einen Zirkel begibt. Diese Maschine bleibt buchstäblich in den Zwischenrechnungen hängen und schreibt daher keine Symbole der ersten Art mehr an. Das bedeutet, dass die Maschine zwar durchaus weiter ihrer Tabelle folgt und operiert, sie aber keine Ziffern ‚0' und ‚1' für die zu berechnende Zahl mehr ausgibt. Da es für Maschinen, die in solch einen Zirkel geraten, kein Herauskommen mehr gibt, und zwar völlig unerheblich davon, wie lange sie an der entsprechenden Zwischenrechnung weiterarbeiten mögen, können diese auch nur eine endliche Anzahl von Ziffern als Ergebnis ausgeben. Ist eine Maschine dagegen *zirkelfrei*, so gelingt es ihr, aus jeder Zwischenrechnung nach einer endlichen Anzahl von Schritten wieder herauszugelangen und ein Symbol der ersten Art auf das Band zu schreiben. Dadurch geben zirkelfreie Maschinen stets eine unendliche Anzahl von Symbolen der ersten Art aus.[35]

Angesichts dieser Definition von zirkulären und zirkelfreien Maschinen könnte man jene Maschinen vermissen, die zwar in keiner zirkulären Zwischenrechnung hängenbleiben, dafür aber in ihrer Tabelle einen definierten Endzustand haben, nach dessen Erreichen sie halten. Diese Turingmaschinen wären zirkelfrei, gäben aber wie zirkuläre Maschinen nur eine endliche Anzahl von Ziffern aus. Da Turing aber kurzerhand die endlichen Zahlen *nicht* zu den berechenbaren zählt, kann er diese Maschinen für seine Argumentation vernachlässigen.[36]

[34] Vgl. Copeland 2004, 33.
[35] Vgl. ebd.
[36] Vgl. Turing 1936, 58 sowie Copeland 2004, 33.

2.2.1.3 Der Hintergrund des Problems

Mit diesen beiden speziellen Definitionen Turings im Hinterkopf kann das *Satisfactoriness Problem* dargestellt werden:[37]

Ausgangsbasis der Überlegungen Turings ist dabei die Vorwegnahme einer bestimmten Kritik an seiner Behauptung, die Menge der berechenbaren Zahlen sei abzählbar und nicht überabzählbar unendlich groß. Eingewandt werden könnte nämlich, dass sich mittels des *Diagonalverfahrens* eine weitere Zahl konstruieren ließe, welche die Menge der berechenbaren Zahlen sprengen würde. Dieses Diagonalverfahren, das auf Georg Cantor und seine Begründung der Mengenlehre zurückgeht, sieht dabei zu Beginn Folgendes vor:
Angenommen, alle berechenbaren Zahlen seien als Binärzahlen in einer Liste aufgeschrieben worden, zum Beispiel von einer unendlichen Maschine. Ihrer Definition gemäß haben diese Zahlen einen unendlichen Dezimalausdruck, so dass sie sich nach rechts in das Unendliche weiter erstrecken. Da es sich außerdem um eine abzählbar unendlich lange Liste von berechenbaren Zahlen handelt, erstreckt sich diese Liste auch nach unten ins Unendliche:

```
0110010101100010010110100100011110 1 …
0101110100111000111111111111110111 …
1101000001101101010000011001000001 1 …
                    .
                    .
                    .
```

Liste der abzählbar unendlich vielen berechenbaren Zahlen.[38]

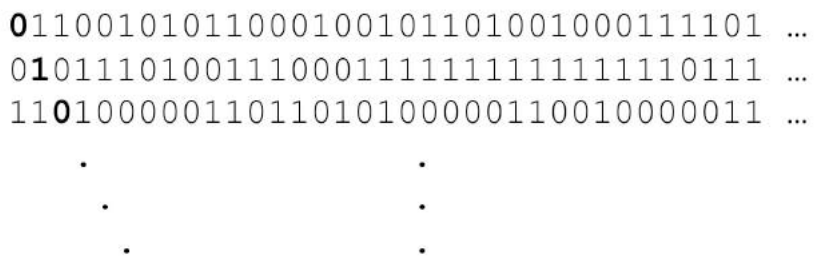
In dieser Liste werden die berechenbaren Zahlen wie folgt angeordnet: In der ersten Zeile wird das Ergebnis der zirkelfreien Turingmaschine mit der kleinsten *description number* geführt. In der zweiten Zeile folgt entsprechend die Zahl, die von der zirkelfreien Turingmaschine mit der zweitkleinsten *description number* ausgegeben wird, usw., in aufsteigender Reihenfolge. Dies stellt sicher, dass jede mögliche berechenbare Zahl in dieser Liste vorhanden ist. Einige werden sogar doppelt vorhanden sein, da es sein kann, dass mehrere Turingmaschinen mit verschiedener *description number* die selbe Zahl ausgeben.

[37] Die Beschreibung orientiert sich an Turing 1936, 72f. und Copeland 2004, 33-39.
[38] Die Abbildung orientiert sich an Copeland 2004, 34. Strenggenommen handelt es sich nach Turings Terminologie bei den Ziffernfolgen in dieser Liste nicht um berechenbare Zahlen, sondern um berechenbare Folgen. Die vorliegende Arbeit verzichtet auf diese Unterscheidung, die bei Turing ohnehin im Wesentlichen nur vom Setzen eines zusätzlichen Dezimalpunkts vor der Folge abhängt. Vgl. Turing 1936, 60.

Copeland behauptet nun Folgendes: Wenn sich zusätzlich zu dieser abzählbar unendlichen Liste der berechenbaren Zahlen eine weitere Zahl konstruieren ließe, die nachweislich noch nicht auf der Liste steht, dann wäre damit bewiesen, dass es nur abzählbar viele berechenbare Zahlen geben kann: „To prove that not all infinite binary sequences are computable, it is enough to describe one that does not appear on this list.“[39] Solch eine Zahl ließe sich mittels des Diagonalverfahrens erzeugen.[40] Diese Haltung mag aus einer globaleren Perspektive auf das Problem ihre Berechtigung haben. Fakt ist aber, dass Turing das Diagonalverfahren aus dem genau entgegengesetzten Grund anführt, nämlich als Basis für die oben genannte Kritik, die zeigen will, dass die Menge der berechenbaren Zahlen eben nicht abzählbar, sondern überabzählbar unendlich sei. In Turings Worten:

> It may be thought that arguments which prove that the real numbers are not enumerable would also prove that the computable numbers and sequences cannot be enumerable. […] [W]e might apply the diagonal process. "If the computable sequences are enumerable, let α_n be the n-th computable sequence, and let $\Phi_n(m)$ be the m-th figure in α_n. Let β be the sequence with 1 - $\Phi_n(n)$ as its n-th figure. Since β is computable, there exists a number K such that 1 - $\Phi_n(n) = \Phi_K(n)$ all n. Putting $n = K$, we have $1 = 2\Phi_K(K)$, *i.e.* 1 es even. This is impossible. The computable sequences are therefore not enumerable."[41]

Tatsächlich lässt sich mittels des Diagonalverfahrens, das Turing hier mathematisch angegeben hat, diese zusätzliche Zahl β konstruieren. Prämisse ist dabei, dass die Zahl β sich von jeder Zahl in der Liste der berechenbaren Zahlen unterscheidet, da sie sonst schon in dieser vorhanden wäre. Dies wird sehr einfach gewährleistet: Wenn die erste Ziffer der ersten Zahl in der Liste eine 1 ist, dann wird die erste Ziffer von β eine 0 sein. Ist die erste Ziffer der ersten Zahl dagegen eine 0, so wird die erste Ziffer von β eine 1 sein. Damit ist klar, dass die Zahl β in jedem Fall nicht mit der ersten Zahl auf der Liste identisch sein kann, da sie sich mindestens in der ersten Stelle von ihr unterscheidet. Ebenso wird die zweite Ziffer der Zahl β in Bezug zur zweiten Ziffer der zweiten Zahl der Liste bestimmt, die dritte Ziffer in Bezug zur dritten Ziffer der dritten Zahl, usw. Turing drückt dies in der Formel 1 - $\Phi_n(n)$ aus.



Für das Diagonalverfahren wird jeweils die n-te Ziffer der n-ten Zahl herangezogen. So bildet sich eine Diagonale quer über die unendliche Liste.

[39] Copeland 2004, 34.
[40] Vgl. ebd.
[41] Turing 1936, 72.

Auf diese Weise ist sichergestellt, dass die Zahl β mit keiner der berechenbaren Zahlen auf der Liste übereinstimmen kann, da sie sich von jeder dieser Zahlen *n* mindestens an der *n*-ten Stelle unterscheiden muss. Folglich lässt sich mittels des Diagonalverfahrens tatsächlich eine Zahl erzeugen, die noch nicht auf der Liste steht. Es hat damit den Anschein, als wäre die Kritik berechtigt und als könnte die Menge der berechenbaren Zahlen nicht abzählbar unendlich sein, sondern müsste, ganz wie die Menge der reellen Zahlen, überabzählbar unendlich sein. Das aber wäre fatal für Turings Ansatz.[42]

Das Diagonalverfahren ist für Turing in seiner mathematischen Gültigkeit nicht zu bezweifeln, so dass die Existenz[43] der Zahl β nicht einfach geleugnet werden kann. Was Turing daher mit dem *Satisfactoriness Problem* zu beweisen sucht, ist, dass die Zahl β zwar existiert, aber dass es sich dabei um *keine berechenbare Zahl* handelt.

2.2.1.4 Turings Reduktion des Problems

Der erste Schritt Turings ist es, das Problem zu reduzieren. Wie gezeigt wurde, lässt sich die Zahl β auf Basis der Liste der berechenbaren Zahlen erzeugen, indem die *n*-te Stelle der *n*-ten Zahl auf der Liste als *n*-te Stelle von β herangezogen wird, nur dass für β noch jede der Stellen umgeschaltet wird, indem anstelle jeder 1 eine 0 und anstelle jeder 0 eine 1 gesetzt wird. Für diesen letzten Schritt lässt sich leicht eine Turingmaschine angeben, die beispielsweise die folgende Form haben könnte:

State	Scanned Square	Operations	Next State
a	0	P[1], R	a
a	1	P[0], R	a

Tabelle einer Turingmaschine, die sämtliche Stellen einer existierenden unendlichen Binärzahl umschaltet.[44]

Das Problem, β zu erzeugen, lässt sich darum auf die simplere Frage reduzieren, ob die Zahl, die sich durch den ersten Schritt der Diagonalisierung in der Liste der berechenbaren Zahlen erzeugen lässt, berechenbar ist. Diese Zahl, die Turing β' nennt, entsteht also wie die Zahl β, nur dass der letzte Schritt mit der Umschaltung aller Stellen nicht durchgeführt wird. Die binäre Zahl β' unterscheidet sich folgerichtig an jeder einzelnen Stelle um den Wert 1 von β.

[42] Allein schon deswegen, weil die nur abzählbar unendlich vielen *description numbers* nicht ausreichen würden, um die laut diesem Ergebnis überabzählbar unendlich vielen berechenbaren Zahlen abzudecken. Das Konzept der Turingmaschine wäre damit mehr als fragwürdig.
[43] Über die tatsächliche Form der Existenz von Zahlen wird in dieser Arbeit kein Urteil getroffen.
[44] Die Terminologie orientiert sich an *Table 1* aus Kapitel 2.1. Vgl. Copeland 2004, 8.

$$\beta = 10100100111010100111100 \ldots$$
$$\beta' = 01011011000101011000 11 \ldots$$

Die Zahl β' entsteht direkt aus dem Diagonalverfahren und unterscheidet sich in jeder Stelle von β.[45]

Da wie oben gezeigt eine Turingmaschine existiert, die durch Umschaltung aller Ziffern aus β' die Zahl β erzeugen kann, gilt: Wenn sich β' als unberechenbar erweisen sollte, dann muss laut Turings Maschinendefinition auch β unberechenbar sein. Für Turings Ziel reicht es also, zu zeigen, dass β' nicht berechenbar ist.

2.2.1.5 Die Unberechenbarkeit des Satisfactoriness Problems

Das *Satisfactoriness Problem*, das Turing für diesen Zweck präsentiert, setzt allerdings an einem zuerst einmal unerwarteten Punkt ein: Turing unterteilt die *standard descriptions* bzw. *description numbers* in solche, die befriedigend sind (*satisfactory*), und solche, die unbefriedigend sind (*unsatisfactory*). Die *standard description* bzw. *description number* einer Turingmaschine gilt als *befriedigend*, wenn sie zu einer zirkelfreien Maschine gehört. Andernfalls, wenn sie zu einer zirkulären Maschine[46] gehört, gilt sie als *unbefriedigend*.[47] Diese Unterscheidung ist die für das *Satisfactoriness Problem* wesentliche Definition, wie im Folgenden deutlicher werden wird.

Um zu klären, ob β' und damit auch β berechenbar ist, nimmt Turing Folgendes an: „Let us suppose […] that we can invent a machine *D* which, when supplied with the S.D of any computing machine *M* will test this S.D and if *M* is circular will mark the S.D with the symobl "*u*" and if it is circle-free will mark it with "*s*"."[48] Die Maschine *D* wäre damit in der Lage, diejenigen *description numbers* zu finden, die zu den oben in der Liste versammelten berechenbaren Zahlen gehören. Da die Maschine *D* so alle zirkelfreien Turingmaschinen angeben könnte, deren Ergebnisse diese Liste bilden, müssten die *description numbers* dieser Turingmaschinen nur noch wie für die Liste vorgesehen der Größe nach geordnet werden, und schon könnte die universale Turingmaschine die Zahl β' berechnen, indem sie die *n*-te Turingmaschine bis zur *n*-ten Stelle simuliert und die sich dort ergebende Ziffer für die *n*-te Stelle von β' notiert – also schlicht das Diagonalverfahren durchführt.

[45] Die angegebenen Zahlenfolgen sind natürlich willkürliche Beispiele rein zu Zwecken der Illustration.
[46] Auch die *standard descriptions*, die nur eine endliche Folge von Ziffern ausgeben oder die gar keine syntaktisch korrekten Maschinen definieren, gelten als unbefriedigend.
[47] Vgl. Turing 1936, 68. Auch diese Terminologie Turings zeigte sich nicht als über spätere Kritik erhaben: Vgl. Copeland 2004, 36.
[48] Turing 1936, 72f. Die Maschinen *D* und *M* wurden im Zitat mittels der moderneren Entsprechungen der bei Turing verwendeten Buchstaben benannt.

Damit ist der Zusammenhang zwischen *Satisfactoriness Problem* und der Berechnung der Zahlen β' und β klar: Wenn die Maschine *D* existiert und sie alle befriedigenden *description numbers* versammeln kann, dann ist auch β' berechenbar – und umgekehrt.
Folgerichtig geht es Turing im *Satisfactoriness Problem* darum, zu zeigen, dass es die Maschine *D* nicht geben kann. Hierfür führt Turing eine *reductio ad absurdum* durch, indem er annimmt, die Maschine *D* existiere, und daraus einen logischen Widerspruch ableitet. Auch diese Argumentation vollzieht Turing strikt an Maschinenmodellen.

Angenommen also, es gäbe die Maschine *D*. Dann wäre es nach dem auf der vorhergehenden Seite vorgestellten Gedanken ein Leichtes, diese Maschine *D* mit der universalen Turingmaschine *U* zu kombinieren. Die dabei entstehende neue Maschine definiert Turing als die Maschine *H*.
Weiterhin unterteilt Turing die Bewegung der Maschine *H* in Abschnitte (*sections*). In jedem Abschnitt wird geprüft, ob eine bestimmte ganze Zahl einer befriedigenden *description number* entspricht. In den ersten *N* -1 Abschnitten testet die Maschine *D*, die nun ja Teil von *H* ist, dementsprechend die ganzen Zahlen 1, 2, …, *N -1*. Entspricht die geprüfte ganze Zahl einer befriedigenden *description number*, dann simuliert *H* die entsprechende Turingmaschine bis zur *n*-ten Stelle und trägt die Ziffer dieser *n*-ten Stelle als neue Ziffer für β' auf dem Band ein.[49] Wenn *H* mit den Abschnitten $N - 1$ fertig ist, testet die Maschine die *description number N*. Wenn *N* befriedigend ist, ist auch die Turingmaschine *N* zirkelfrei, so dass sie von *H* simuliert werden kann, und auch hier die *n*-te Ziffer der von dieser Maschine *N* ausgegebenen Zahl auf dem Band vermerkt wird. Entspricht die Zahl *N* keiner zirkelfreien Maschine, rückt die Maschine *H* zum nächsten Abschnitt vor und testet die ganze Zahl $N + 1$.
Diese detaillierte Beschreibung Turings dient einem einfachen Zweck: Sie zeigt, dass die Maschine *H* stets von einem Abschnitt zum nächsten voranschreitet. Dabei gibt sie zwar nicht nach jedem Abschnitt, aber *immer wieder* eine Ziffer für β' aus, nämlich immer dann, wenn die getestete ganze Zahl einer befriedigenden *description number* entspricht. Dieses *immer wieder* bedeutet, dass die Maschine *H* unendlich viele Symbole der ersten Art ausgibt. Damit ist klar, dass *H* selbst eine *zirkelfreie Maschine* sein muss.[50]
Was aber passiert, wenn *H*, die ja alle nur möglichen Turingmaschinen prüft, auf ihre eigene *description number* – in Turings Terminologie die *description number K* – stößt? Eine berechtigte Frage, denn irgendwann muss dies unweigerlich der Fall sein.

[49] Dabei ist zu beachten: Das *n* für die *n*-te Stelle entspricht nicht der von *D* getesteten ganzen Zahl, sondern bezieht sich darauf, wie viele Turingmaschinen von *D* bereits als zirkelfrei befunden wurden.
[50] Vgl. ebd., 73.

Würde die Maschine *H* auf ihre eigene *description number K* stoßen – und dies ist das Finale von Turings *Satisfactoriness Problem* – dann würde sich Folgendes ereignen:
Wie bei allen anderen *description numbers* auch würde die Maschine *D*, die ja Teil der Maschine *H* ist, prüfen, ob es sich bei *K* um eine befriedigende oder um eine unbefriedigende Beschreibungsnummer handelt. Nun muss es sich bei *K* um eine befriedigende Beschreibungsnummer handeln, schlicht deswegen, weil die Untersuchung der Funktionsweise der Maschine *H* gezeigt hat, dass diese zirkelfrei ist.
D erkennt die *description number K* also als befriedigend. Daher hat *H* die durch *K* definierte Turingmaschine auszuführen. Das bedeutet aber, dass *H* ihr eigenes Verhalten simulieren muss. Entsprechend beginnt *H* erneut damit, die Turingmaschinen beginnend bei den Beschreibungsnummern 1, 2, … zu simulieren – denn dies ist es eben, was *H* tun muss, ob simuliert oder nicht. Das geht auch eine Zeit lang gut, bis *H*'s Simulation ihrer selbst erneut bei der Zahl *K* angelangt ist und sie jetzt eigentlich die neue Ziffer für β' berechnen sollte. Dazu kommt es aber nicht. Die von *H* ausgeführte Maschine mag nur simuliert sein, nichtsdestoweniger handelt es sich bei der simulierten Maschine eben auch um *H*. Somit hat auch die simulierte Maschine *H* ab hier keine andere Wahl, als die Turingmaschine mit der *description number K* auszuführen, was heißt, dass die simulierte Maschine ebenfalls dazu übergehen muss, sich selbst zu simulieren. Entsprechend muss die echte *H*, nun in einer doppelten Simulation ihrer selbst, erneut ihr eigenes Verhalten von vorne nachvollziehen und beginnt wieder mit dem Prüfen und Ausführen der ersten Beschreibungsnummern von 1, 2, … an. Diese rekursive Bewegung hat keinerlei Endpunkt, so dass *H* bodenlos in einer Art *Mise en abyme* versinkt und die in *K* gesuchte nächste Ziffer der Zahl β' nie ausgegeben wird.

Künstlerische Umsetzung einer *Mise en Abyme*, eine ideal gesehen endlose Folge von Bild im Bild.[51]

[51] Abbildung von Benvenuto 2015 (Internetquelle).

Nun könnte man meinen, dies sei schon die ganze Bedeutung des *Satisfactoriness Problems*: Da die Maschine *H* sich in einer endlosen Rekursion verliert, kann β' eben nicht berechnet werden. Dies wäre aber zu kurz gedacht. Es geht in Turings Beweis nicht darum, dass die Turingmaschine im Falle von β' endlos rechnen müsste. Wie diese Arbeit noch zeigen wird, sind Maschinen denkbar, die auch unendlich viele Arbeitsschritte in endlicher Zeit ausführen. In der Konstruktion von *H* verbirgt sich allerdings ein logischer Widerspruch. Dieser logische Widerspruch, der Turings *reductio ad absurdum* vollständig macht und damit zeigt, dass die Maschine *D* logisch unmöglich ist und weder die Zahl β' noch die Zahl β berechenbar sind, steckt im Detail:

Wie Turing definierte, sind nur die Ziffern ‚0' und ‚1', die das Ergebnis, d. h. die von einer Turingmaschine berechnete Zahl bilden, Symbole erster Ordnung. Wenn die Maschine *H* bei der Ermittlung der nächsten Ziffer von β' eine Turingmaschine simuliert, dann schreibt sie daher eben keine Symbole dieser ersten Ordnung, sondern führt die Simulation mit Symbolen der zweiten Ordnung auf den *E-Feldern* als Zwischenrechnung durch. *H* gibt also erst wieder ein Symbol der ersten Ordnung aus, wenn die Simulation der aktuellen Turingmaschine abgeschlossen, die nächste Ziffer für β' ermittelt ist und die entsprechende ‚0' oder ‚1' auf das nächste *F-Feld* geschrieben wird.

Diese Distinktion von Symbolen erster und zweiter Art hat fatale Folgen für die Konzeption von *H*. Sie bedeutet, dass auch *H* nur Symbole der zweiten Art schreibt, wenn sie sich selbst simuliert. Da *H* bei der Selbstsimulation jedoch unweigerlich in die endlose *Mise en abyme* gerät, wird die Maschine ab hier kein Symbol der ersten Art mehr ausgeben. Das aber heißt, dass *H* nur eine endliche Anzahl von Symbolen der ersten Art ausgibt und somit per Definition eine zirkuläre Maschine ist. Dies widerspricht jedoch dem Urteil, dass *H* zirkelfrei ist. Da *H* nicht zugleich zirkelfrei und zirkulär sein kann, muss daraus geschlossen werden, dass es *H* nicht geben kann.[52]

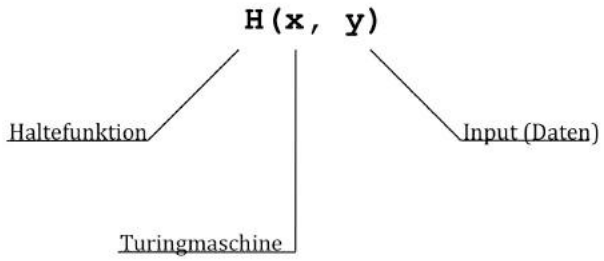
Wo aber liegt der logische Fehler in *H*? Weder die in *H* eingebaute universale Turingmaschine *U* noch die Kombination der Maschinen *D* und *U* stellen in dieser Hinsicht vor Schwierigkeiten. Es bleibt daher nur der Schluss, dass die Maschine *D* logisch unmöglich ist. Es kann also keine Maschine geben, die für eine beliebige ganze Zahl entscheidet, ob es sich bei dieser um eine befriedigende *description number* handelt. Somit kann auch die Liste der berechenbaren Zahlen nicht von einer Maschine erstellt werden. Damit hat Turings *reductio ad absurdum* logisch erwiesen, dass die Zahlen β' wie auch β nicht zu den berechenbaren Zahlen gehören, sondern vielmehr unberechenbare Zahlen sind.

[52] Vgl. Turing 1936, 73.

2.2.2 Das Halteproblem

Eine weitere unberechenbare Funktion ist das moderne Halteproblem. Dieses stammt im Gegensatz zum *Satisfactoriness Problem* nicht von Turing selbst, sondern hat seinen Namen von Martin Davis, der es allem Anschein nach auch als erster entwickelt hat. Nichtsdestoweniger wird das Halteproblem gerne mit ‚On Computable Numbers' in Verbindung gebracht. In den allermeisten neueren Arbeiten wird es der Einfachheit halber anstelle des *Satisfactoriness Problem* behandelt. Tatsächlich hat das Halteproblem auch große Ähnlichkeit zum hier nicht weiter thematisierten *Printing Problem* Turings. Strenggenommen aber wurde es in ‚On Computable Numbers' weder benannt noch bewiesen.

Auch im Halteproblem geht es um die Frage, ob eine Turingmaschine, die von einer willkürlich gewählten *description number* definiert wird, irgendwann anhält oder unendlich weiter läuft. Im Gegensatz zum *Satisfactoriness Problem* aber geht es hier nicht um zirkuläre und zirkelfreie Maschinen, sondern darum, ob die Turingmaschine insgesamt anhält, d. h. ob der Scanner nach endlich vielen Operationsschritten über einem Kästchen zum Halten kommt und die gesamte Maschine ihre Arbeit einstellt.[53] Die dieses Halteproblem definierende Haltefunktion erhält als Input nacheinander die Beschreibungsnummern aller Turingmaschinen (x) und alle möglichen Inputs für diese Maschinen (y).[54]



Schematische Darstellung der Haltefunktion.

Die Funktion H prüft für alle Turingmaschinen x mit Input y, ob sie anhalten, oder nicht. Dies ist gleichbedeutend mit der Frage, ob ein bestimmter Algorithmus terminiert. Das sogenannte Haltetheorem besagt, dass es keine systematische Methode, d. h. auch keine Turingmaschine geben kann, die das Halteproblem löst. Dieser Umstand ist der Grund dafür, dass reale Computer über keine allgemeingültige Warnanzeige verfügen können, die den Nutzer vom Ausführen derjenigen Programme abhält, die zum Aufhängen des Computers führen.

[53] Vgl. Copeland 2004, 39.

[54] In dieser Darstellung wird der heute oftmals verwendeten Trennung zwischen Turingmaschine und ihrem Input Rechnung getragen, diese ist für Turings Modell allerdings nicht notwendig. Da sich Turingmaschine und Input zu einer neuen Turingmaschine mit eigener *description number* zusammenfassen lassen, wird diese Trennung nicht weiter expliziert.

Auch wenn es keine Maschine geben kann, welche die Haltefunktion H ausführt, lässt sich diese Funktion dennoch für die Theorie weiter spezifizieren, indem festgelegt wird, dass sie eine ‚0' ausgeben soll, wenn die aktuell geprüfte Turingmaschine nicht terminieren wird, und eine ‚1', wenn sie anhält. Daraus lässt sich zum einen eine spezielle unberechenbare Menge bilden, und zwar die Menge aller Turingmaschine, die anhalten: Die Haltemenge (*halting set*). Zum anderen lässt sich hieraus auch eine unberechenbare Zahl ableiten. Hierfür werden die Ergebnisse der spezifizierten Haltefunktion schlicht wie im Fall von β' zu einer unendlichen Binärzahl konkateniert. Der ersten Stelle der Zahl wird eine ‚1' zugewiesen, wenn die erste Turingmaschine in der Menge aller Turingmaschinen anhält, und eine ‚0', wenn nicht. Entsprechend wird an der *n*-ten Stelle der Zahl eine ‚1' gesetzt, wenn die *n*-te Turingmaschine in der Menge aller Turingmaschinen anhält, und eine ‚0', wenn nicht. Die sich so ergebende unendliche und aufgrund des Haltetheorems unberechenbare Binärzahl wird mit dem Symbol τ für Turingkonstante bezeichnet.[55]

2.3 Von Zahlen zu Maschinen

Wie die Darstellung des *Satisfactoriness Problems* und des Halteproblems gezeigt hat, existieren tatsächlich Zahlen, die sich zwar definieren lassen, aber nicht berechnet werden können. Dabei ist klar geworden, dass die Menge der berechenbaren Zahlen gegenüber der Menge der unberechenbaren Zahlen tatsächlich sehr klein ausfällt. Festgestellt wurde somit auch, dass der Turingmaschine klare Grenzen bezüglich dessen gesetzt sind, was mit ihr generiert werden kann.

Der Mathematiker und Logiker Kurt Gödel lobt die Turingmaschine dementsprechend auch als die klarste Definition von Berechenbarkeit überhaupt. Seine Lesart versteht die Turingmaschine in einem anders nuancierten Licht: Für ihn definieren ihre Grenzen das, was mit Maschinen überhaupt möglich ist. Der zentrale Begriff, den Gödel dabei verwendet, und der für ihn mit der Turingmaschine synonym ist, ist der Begriff der *mechanischen Prozedur*. Gödel vollzieht diesen Übergang von Zahlen zu Maschinen, ohne dabei gleich im Sinne der erweiterten Church-Turing These Aussagen über die Struktur des Universums treffen zu wollen. Seine Lesart der Turingmaschine bietet sich daher in besonderer Weise an, wenn diese für medientheoretische Erkenntnis fruchtbar gemacht werden soll.

[55] Vgl. Ord 2002, 5. Es wäre von großem Interesse, für die noch folgenden Gedanken in dieser Arbeit auch die Erkenntnisse der modernen Informationstheorie heranzuziehen. Besonders die Analyse von Chaitins Ω-Zahl (vgl. Chaitin 1975, 1990 und 2000) und des Umstands, dass τ aus dem Blickwinkel der Informationstheorie zumindest *semi-computable* ist (vgl. Ord 2002, 5), würde die Untersuchung auf breiteren Fuß stellen.

3. Die mechanische Prozedur. Gödels Lesart der Turingmaschine

Der Mathematiker, Logiker und sicherlich auch Philosoph[56] Kurt Gödel (1906-1978) ist vor allem für seine beiden Unvollständigkeitssätze bekannt, mit denen er zeigte, dass das Hilbertprogramm, also die vollständige Axiomatisierung der Mathematik, nicht erfüllt werden kann. Gödel bewies in seiner 1931 veröffentlichten Arbeit, dass es im System der *Principia Mathematica* und in verwandten Systemen zwangsläufig wahre Sätze geben muss, die nicht beweisbar sind, d. h. wahre Sätze *A*, für die *unentscheidbar* ist, ob *A* oder ~*A* gilt.[57] Für David Hilbert selbst und für all diejenigen, die ihre Hoffnungen in das von diesem maßgeblich bestimmte Projekt der Grundlagensuche in der Mathematik gesetzt hatten, war die Beweisführung Gödels ein vernichtender Schlag, zeigte sie doch, dass die Mathematik mit den von Hilbert geforderten finiten Mitteln unmöglich zu fundieren war. Andere, wie der Mathematiker und später rüstungstechnisch involvierte[58] Johann von Neumann, hatten die Existenz solcher unentscheidbaren Sätze in formalen Systemen bereits vorausgeahnt. So schrieb von Neumann schon im Jahr 1927 zur Beweistheorie Hilberts:

> […] [D]ie Unentscheidbarkeit ist […] die Conditio sine qua non dafür, daß es überhaupt einen Sinn habe, mit den heutigen heuristischen Methoden Mathematik zu treiben. An dem Tage, an dem die Unentscheidbarkeit aufhörte, würde auch die Mathematik im heutigen Sinne aufhören zu existieren; an ihre Stelle würde eine absolut mechanische Vorschrift treten, mit deren Hilfe jedermann von jeder gegebenen Aussage entscheiden könnte, ob diese bewiesen werden kann oder nicht.[59]

Von Neumanns Sätze erhellen aufs Deutlichste die Implikationen, die ein Erfolg des Hilbertprogramms nach sich gezogen hätte: Wäre die Mathematik in der von Hilbert und anderen gestalteten formalisierten Ausprägung aus sich selbst heraus als widerspruchsfrei und vollständig bewiesen worden, dann könnte von da an jede Person, ganz gleich, ob sie etwas von Mathematik verstünde oder nicht, für jede mathematische Aussage entscheiden, ob sie bewiesen werden kann. Dafür müsste diese Person, ganz wie im vorhergehenden Kapitel beschrieben, lediglich wie die Turingmaschine einer „absolut mechanischen Vorschrift" folgen. Es wäre also keinerlei Verständnis von Sinn oder Bedeutung dieser Prozedur über die elementaren mechanischen Schritte hinaus erforderlich. Ein Erfolg des Hilbertprogramms

[56] Die *Collected Works* zu Gödel beinhalten einige seiner philosophischen Überlegungen über die Mathematik. Eine empfehlenswerte Quelle ist in diesem Kontext aber auch Hao Wang (besonders Wang 1987). Zu bemerken ist in diesem Kontext auch die *Kurt Gödel Forschungsstelle* zu den „Philosophischen Bemerkungen" Gödels, die 2015 an der Berlin-Brandenburgischen Akademie der Wissenschaften eingerichtet wurde, und die sich speziell den bisher noch nicht wissenschaftlich aufbereiteten philosophischen Beiträgen Gödels widmen wird (vgl. Berlin-Brandenburgische Akademie der Wissenschaften 2015, 150).
[57] Vgl. dazu Gödel 1931, aber auch Nagel/Newman 1958.
[58] Zu von Neumanns Tätigkeit als „Falke" vgl. Hagen 2002.
[59] Neumann 1927, 12.

hätte also nichts weniger bedeutet, als die Möglichkeit, jede Entscheidung über die Beweisbarkeit von mathematischen Formeln an eine Maschine abzugeben. Diese Maschine könnte dann, bei entsprechender Bauweise, *die gesamte wahre Mathematik nach rein mechanischen Gesetzen hervorbringen.*

Dieses Programm einer vollständigen Mechanisierung der Mathematik hat Gödel als unmöglich bewiesen. Dass daraufhin Turing die Ergebnisse Gödels als wesentliche Grundlage nutzte – wie Gödel sich zuvor der *Principia Mathematica* für seine Beweisführung bedient hatte – zeigt nicht nur die Herkunftslinie der modernen Rechner aus dem Grundlagenprojekt der Mathematik,[60] sondern auch, dass der Begriff der Berechenbarkeit und die Computer genealogisch etwas sind, das aus der Erkenntnis der Beschränkungen aller (symbolischen) Maschinen erwachsen ist: *Die Turingmaschine ist eine Maschine der Grenze.*

3.1 Maschinen statt Menschen

Vor diesem Hintergrund nimmt es nicht Wunder, dass sich in Alan Turings Arbeiten direkte Bezugnahmen auf Gödel finden: So als abgrenzende Verweise in ‚On Computable Numbers'[61], als explizite Anführung in ‚Intelligente Maschinen, eine häretische Theorie'[62], in ‚Systems of Logic Based on Ordinals'[63] und ‚Solvable and Unsolvable Problems'[64], sowie ohne namentliche Nennung, aber mehr als deutlich, am Ende von ‚The State of the Art'[65].

Kurt Gödel wiederum bezieht sich ebenso auf Turing. Von ungefähr 1938 an – der betreffende Text Gödels lässt sich nicht genau datieren[66] – rekurriert er auf Turings Arbeit als „establishing the "correct definition" of mechanical computability", eine Auffassung, bei der er bleiben wird. So schreibt Gödel 1964: „Turing's work gives an analysis of the concept of "mechanical procedure" […]. This concept is shown to be equivalent with that of a "Turing machine."“[67] Es lassen sich drei Argumente anführen, dass Gödel, obwohl er als Mathematiker und Logiker aus eben diesem Kontext heraus argumentiert, die Turingmaschine als tatsächliche Maschine auslegt, dass also der Terminus ‚mechanisch' Konnotationen aufweist, die über eine rein mathematische Bedeutung hinausgehen.

---

[60] Vgl. dazu besonders Heintz 1993.
[61] Vgl. Turing 1936, 59 und 84.
[62] Vgl. Turing 1951, 9.
[63] Vgl. Turing 1939, 161.
[64] Vgl. Turing 1954, 593.
[65] Vgl. Turing 1947, 206.
[66] Gödel 193? [sic].
[67] Gödel 1964, 369f. Vgl. dazu auch: Gödel 193?, 166 und 168 sowie Gödel 1951, 304f. Weiter die Fußnote vom 28. August 1963 zu Gödel 1931 sowie Shagrir 2006, 8.

(1) Nicht nur wird der genannte Begriff der mechanical procedure – der mechanischen Prozedur – von Gödel zentral verwendet, wenn es um die Turingmaschine geht, sodass die mechanische Prozedur für Gödel äquivalent ist zu dem, was anhand der Turingmaschine analysiert wird bzw. zur Turingmaschine selbst. Die mechanische Prozedur ist auch äquivalent zu „algorithm", zu „computation procedure", zu „finite combinatorial procedure",[68] zu „rule of inference [in a formal system]", zu „mechanical computability",[69] zu „formal system" bzw. zu „any mechanical procedure for producing formulas, called provable formulas",[70] zu „finite procedure"[71] und zu „a machine with a finite number of parts"[72]. Dabei ist es die letztgenannte Formulierung der „machine with a finite number of parts", die den Begriff mechanische Prozedur vor allem anderen in Richtung realer Maschinen rückt. Gödel schreibt im Detail um 1938:

> He [Turing] has shown that the computable functions defined in this way [by Turing's analysis] are exactly those for which you can construct a machine with a finite number of parts which will do the following thing. If you write down any number $n_1, \dots, n_r$ on a slip of paper and put the slip into the machine and turn the crank, then after a finite number of turns the machine will stop and the value of the function for the argument $n_1, \dots, n_r$ will be printed on the paper.[73]

Gödel schreibt diese Sätze zu einer Zeit, zu der er – so Copeland und Shagrir – noch nicht klar zwischen mechanischer und finiter Prozedur unterscheidet.[74] Ein Umstand, der dieser Stelle ihr argumentatives Gewicht nehmen könnte, wenn Gödel nicht die exakt gleiche Formulierung von der „machine with a finite number of parts" 1951 noch einmal anführte:

> The greatest improvement was made possible through the precise definition of the concept of finite procedure, which plays a decisive role in these results. There are several different ways of arriving at such a definition, which however, all lead to exactly the same concept. The most satisfactory way, in my opinion, is that of reducing the concept of finite procedure to that of a machine with a finite number of parts, as has been done by the British mathematician Turing.[75]

Diese Textstellen bieten einen gewissen Spielraum, um Gödels Äquivalenz zwischen der „machine with a finite number of parts" und der Turingmaschine bzw. der mechanischen Prozedur in Beziehung zu endlichen Maschinenmodellen zu setzen. Genügt dies aber, um zu behaupten, Gödel beziehe sich damit auf reale Maschinen? Immerhin ließe sich doch einwenden, dass diese Begriffe lediglich vor dem Hintergrund rein abstrakter Maschinen im Sinne der Turingmaschine ihre Ausprägung erfahren.

---

[68] Gödel 1964, 369f.
[69] Gödel 193?, 166 und 168.
[70] Gödel 1964, 370.
[71] Gödel 1951, 304f. und Gödel 1964, 370.
[72] Gödel 1951, 304f.
[73] Gödel 193?, 168. Vgl. auch: Gödel 1951, 304f.
[74] Vgl. Copeland/Shagrir 2015, 9.
[75] Gödel 1951, 304f.

In den aktuellen Debatten im Feld der Berechenbarkeitstheorie existiert aber keine als zufriedenstellend betrachtete Erklärung für die Formulierung von der „machine with a finite number of parts." So schreibt Wilfried Sieg: „There is no explanation of why such a reduction is the most satisfactory way of getting to a precise definition or, for that matter, of why the concept of a machine with a finite number of parts is equivalent to that of a Turing machine"[76] und weiter: „As above, we can raise the question, why should such machines be Turing machines? Gödel does not answer this question [...]."[77] Auch Copeland und Shagrir können hier nur Vermutungen anstellen – und sie beziehen sich dabei lediglich auf die erstgenannte Textstelle von ungefähr 1938: „Possibly he [Gödel] regarded the concept of a mechanical and finite procedure as somehow captured by the notion of "a machine with a finite number of parts.""[78]
Gäbe es eine befriedigende Erklärung, inwiefern Gödel sich mit seiner Formulierung auf abstrakte Maschinen bezieht, hätte der genannte Einwand einiges Gewicht. Weil die Formulierung von der „machine with a finite number of parts" aber gerade dann Rätsel aufgibt, wenn sie in Bezug auf abstrakte Maschinen verstanden wird, erscheint es zumindest fraglich, ob Gödel die Turingmaschine tatsächlich nur als abstrakte Maschine auslegt.

(2) Die gegenteilige Annahme – dass Gödel die Turingmaschine mit Bezug auf reale Maschinen versteht, sodass der Terminus mechanisch eine Konnotation aufweist, die über seine rein mathematische Bedeutung hinausgeht – wird weiter gestützt durch Gödels „fascination […] with the concrete mechanical realization of logical procedures",[79] wie Sieg sie konstatiert. Copeland und Shagrir sprechen in diesem Zusammenhang gar von einer Faszination für „calculating machines"[80] per se. Diese Faszination Gödels für Rechenmaschinen schlägt an prominenten Stellen in seiner Arbeit durch. So beispielsweise in dem bereits zitierten Text von ungefähr 1938, in dem er die Kurbel als konkretes Bauteil der Maschine und auch deren Handhabung benennt: „If you write down any number $n_1, \ldots, n_r$ on a slip of paper and put the slip into the machine and turn the crank […]."[81] Deutlich wird hier schon der Kontrast zur Maschinenbeschreibung Turings, welcher auf jede Art von Vorrichtungen zur Handhabung verzichtet. Gleichzeitig bleibt aber auch bei Gödel die genaue Funktionsweise der Maschine im Dunkeln.

[76] Sieg 2006, 9.
[77] Ebd., 11.
[78] Copeland/Shagrir 2015, 10.
[79] Sieg 2006, 10.
[80] Copeland/Shagrir 2015, 8.
[81] Gödel 193?, 168. Vgl. auch: Gödel 1951, 304f.

Noch umfangreicher nimmt Gödel 1939 in einem Vortrag in Notre Dame Bezug auf konkrete Bauteile, hier Kurbel, Schreibmaschine und Glocke, um zu verdeutlichen, inwieweit die Kalküle der Aussagenlogik und der Prädikatenlogik mechanisierbar sind:[82]

> […] [I]t would actually be possible to construct a machine which would do the following thing: The supposed machine is to have a crank and whenever you turn the crank once around the machine would write down a tautology of the calculus of predicates and it would write down every existing tautology … if you turn the crank sufficiently often. So this machine would really replace thinking completely as far as deriving of formulas of the calculus of predicates is concerned. It would be a thinking machine in the literal sense of the word. For the calculus of propositions you can do even more. You could construct a machine in form of a typewriter such that if you type down a formula of the calculus of propositions then the machine would ring a bell [if the formula is a tautology] and if it is not it would not. You could do the same thing for the calculus of monadic predicates [Letzte Einfügung in eckigen Klammern im Original].[83]

Im Vergleich dazu sticht auch hier die Abwesenheit aller konkreten Bauteile in Turings Entwurf ins Auge, wenn man vom Papierstreifen absieht – der für Turing nicht zur eigentlichen Maschine gehört – und sich daran erinnert, das der moderne Scanner erst später seine konkrete Realisation gefunden hat. Gödel dagegen hat, wenn er Kurbel, Papierstreifen, Schreibmaschine und Glocke anführt und deren praktische Handhabung beschreibt, konkrete Maschinen vor Augen: Apparate wie Schreibmaschinen und Rechenmaschinen und deren Derivate. Dieser Vorprägung Gödels gilt es bei der Bewertung seiner mechanischen Prozedur Rechnung zu tragen.

(3) Wenn belegt werden soll, dass die Turingmaschine nur als abstrakte Maschine gedacht werden darf, wird gerne angeführt, dass Turings Maschinenmodell allein als Werkzeug zur Analyse eines rechnenden Menschen diente und Turing darüber hinaus keine reale Umsetzung im Sinn gehabt habe.[84]

Dagegen ist es mehr als bezeichnend, dass Gödel nicht nur keinen Bezug auf menschliches Rechnen nimmt,[85] sondern Turing sogar für dessen Annahme kritisiert, einen Menschen analysiert zu haben. Diese Kritik Gödels bringt Webb auf den Punkt, wenn er schreibt, Gödel sei der Meinung gewesen, „that all Turing was really *analyzing* was the concept of "mechanical procedure", but in his *arguments* for the adequacy of his analysis he overstepped

[82] Vgl. Sieg 2006, 10.

[83] Der Vortrag ist zum gegenwärtigen Zeitpunkt unveröffentlicht, die handbeschriebenen Seiten liegen aber öffentlich zugänglich in der Firestone Library der Princeton University. Kopien der entsprechenden Seiten wurden dem Verfasser von Wilfried Sieg zur Ansicht gegeben, wofür diesem auch an dieser Stelle noch einmal ausdrücklich gedankt sein soll. Die zitierte Stelle findet sich auch in Sieg 2006, 10.

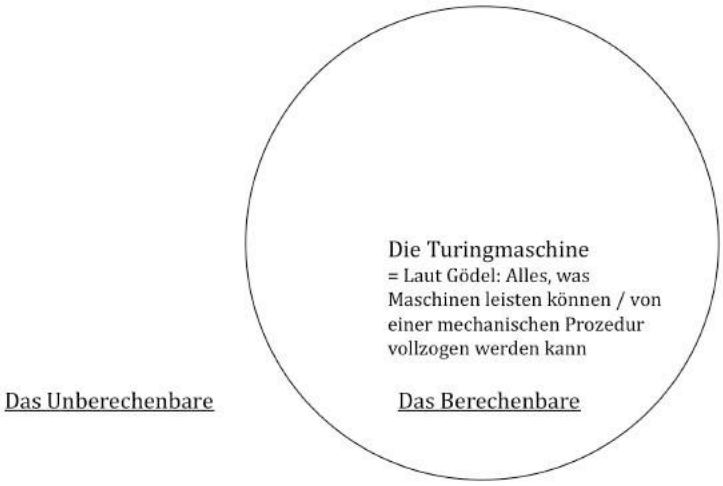
[84] Es ist ohnehin die Frage, inwieweit diese Behauptung Bestand haben kann. So stellt Max Newman klar, dass Turing schon zur Zeit von ‚On Computable Numbers' an eine praktische Umsetzung seiner universalen Maschine dachte, so dass man ihm zumindest keine Unkenntnis der Möglichkeit einer Realisation unterstellen kann. Vgl. hierzu die Diskussion in Copeland 2004, 15f.

[85] Vgl. Copeland/Shagrir 2015, 17.

himself by dragging in the mental life of a human computer.“[86] Auch Copeland und Shagrir stimmen dieser Auffassung Webbs zu: „In our view, Gödel probably regarded Turing’s statements about human cognition as entirely superfluous.“[87] Gödels Lesart der Turingmaschine ist also nicht an das Rechnen als eine menschliche Tätigkeit gekoppelt. Gödel nimmt aber auch nicht an, dass die Turingmaschine etwas Neues in die Welt setze. Das, was diese Maschine definiert – die mechanische Prozedur – existierte vielmehr schon zuvor, wurde aber, wenn man so will, noch nicht klar genug begriffen:

> It is well-known that A. M. Turing has given an elaborate definition of the concept of a mechanically computable function of natural numbers. This definition most certainly was not superfluous. However, if the term "mechanically computable" had not had a clear, although unanalyzed, meaning before, the question as to whether Turing's definition is adequate would be meaningless, while it undoubtedly has an affirmative answer.[88]

In den Augen Gödels ist es die Leistung Turings, das zu definieren, was überhaupt mittels mechanischer Prozeduren berechnet werden kann – der Bezug auf Menschen ist in diesem Kontext dagegen überflüssig, wenn nicht gar fahrlässig. Wie die Formulierung von der „machine with a finite number of parts“ und die drei genannten Argumente zeigen, bezieht sich Gödels Lesart der Turingmaschine stattdessen auf dasjenige Rechnen, das an Maschinen delegiert werden kann und zumindest in Teilen schon immer an solche delegiert wurde. Gödels Auslegung nach ist die Turingmaschine also die Definition dessen, was Maschinen überhaupt leisten können.



Gödels Lesart der Turingmaschine: Diese definiert all das, was Maschinen berechnen können.

[86] Webb in Gödel 1972a, 302.
[87] Copeland/Shagrir 2015, 17.
[88] Gödel 1972, 275n1.

3.2 Vom Berechenbaren zum Unberechenbaren – und zurück

Gödel liefert keine weiteren Erklärungen, wieso die Turingmaschine die Definition dessen sei, was Maschinen „with a finite number of parts“ überhaupt leisten können. Das mag befremdlich anmuten. Schließlich existieren neuere Ansätze, die diese Behauptung nicht einfach setzen, sondern sie detailliert zu beweisen suchen – so beispielsweise Gandys Ansatz von 1980.[89] An diesem Ansatz Gandys zeigt sich zum einen, dass die generelle Behauptung Gödels, die Turingmaschine sei die Definition aller Maschinen, mit symbolischen Mitteln kaum zu begründen ist. Folgerichtig muss Gandy analoge Maschinen ausklammern. Zum anderen ist die Stichhaltigkeit selbst des so eingeschränkten Ansatzes nicht unumstritten.[90]
Wenn aber schon analoge Maschinen aus der Definition der Turingmaschine herausfallen, wie soll Gödels Lesart dann Bestand haben? Tatsächlich überbieten analoge Maschinen *theoretisch* die von der Turingmaschine definierte Grenze der Berechenbarkeit. *Praktisch* aber, und auch das ist eine empirisch nachvollziehbare Beobachtung, existieren keine realen analogen Maschinen, die ein Mehr an Berechenbarkeit bieten würden. Die Turingmaschine in Gödels Lesart rein als symbolisch abgebildete mathematische Papiermaschine verstehen zu wollen, heißt, einen medientechnischen Kategorienfehler zu begehen: Gödel sieht in der Turingmaschine die Definition dessen, was mit *realen* Maschinen überhaupt möglich ist.
Um die Turingmaschine im Sinne Gödels klarer als Maschine zu fassen, soll im Folgenden ein alternativer Weg beschritten werden. Anstatt wie Gandy symbolisierte Maschinen[91] mit der Turingmaschine zu vergleichen, wird der Blick von außerhalb des Berechenbaren, vom Unberechenbaren aus nach innen gerichtet. Hierfür wird im Folgenden zuerst die Grenze der Berechenbarkeit noch einmal empirisch unterfüttert. Daraufhin wird es um die Frage gehen, ob die Abarbeitung unberechenbarer Aufgaben überhaupt logisch möglich ist (Kapitel 4.1). Weiterhin werden einige Modelle vorgestellt, die zumindest konzeptuell auch unberechenbare Funktionen bearbeiten können und in deren reale Umsetzung zum Teil große Hoffnung gesetzt wurde (Kapitel 4.2). Die anhand dieser Modelle gewonnenen Erkenntnisse werden schließlich genutzt, um Implikationen für die Turingmaschine abzuleiten (Kapitel 4.3).

[89] Vgl. Gandy 1980.
[90] Vgl. Copeland/Shagrir 2007.
[91] An dieser Stelle wird die Unterscheidung zwischen symbolisch und symbolisiert wie folgt definiert: Die Turingmaschine und alle digitalen Rechner sind auch operativ *symbolische* Maschinen. Werden diese Computer allerdings diagrammatisch eingefroren, handelt es sich bei diesen nun starren Abbildungen der Computer um *symbolisierte* Maschinen. Mit dieser letzteren Formulierung wird das Moment der Umwandlung vom Operativen zum Erstarrten stark gemacht. Die Maschine wird aus dieser Perspektive weniger als Folge der symbolischen Konstrukte gesehen, sondern umgekehrt, als deren Vorläufer. Da in diese Arbeit aber auch Kittlers Trias des Symbolischen-Realen-Imaginären eingeht, kann die Unterscheidung symbolisch/symbolisiert nicht konsequent durchgehalten werden (s. vor allem die Verwendung in Kapitel 4).

4. Hypercomputation

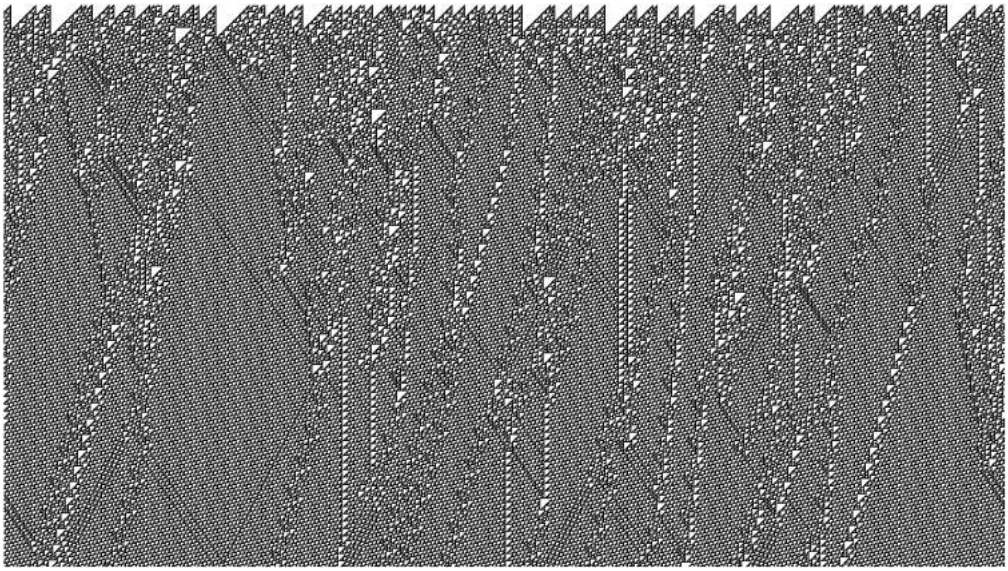
Das empirische Argument für die Turingmaschine als Grenze des für Maschinen Möglichen klang bereits an: Sämtliche heute als reale Maschinen funktionsfähigen Rechnerarchitekturen fallen in den Rahmen dessen, was die Turingmaschine als das Berechenbare aufspannt. Für alle tatsächlich operativen Maschinen gilt also nach wie vor Kittlers Satz aus der Zeit der Jahrtausendwende: „Ein Jenseits der Turingmaschine gibt es bislang gar nicht."[92]

Dabei ist es unerheblich, ob es sich um die Harvard-Architektur mit speichertechnisch getrennter Behandlung von Befehlen und Daten handelt oder um die von Neumannsche Alternative, die diesen Unterschied nivelliert. Selbst Parallelrechner rütteln nicht an der grundlegenden Grenze des mit realen Maschinen Berechenbaren. Es gibt einfach keine Aufgaben, welche die Maschinen der von Neumann-Ära, die ab 1940 begonnen hatte, nicht ebenso lösen können wie die Maschinen der Parallelcomputer-Ära, die von ihren Pionieren ab 1985 ausgeläutet wurde,[93] vielleicht aber wohl erst mit dem Abbruch der Entwicklung der Prozessoren *Tejas* und *Jayhawk* durch Intel im Jahr 2004 ihren Anfang genommen haben mag, als das *frequency scaling* – d. h. die Steigerung der Rechenleistung durch Erhöhung der Taktzahl von einzelnen Prozessoren – aufgrund zu großer Wärmeentwicklung und zu hohem Energieverbrauch an seine Grenzen kam. Woraufhin Intel eben begann, parallel laufende Alternativen zu entwickeln.[94] Zugegeben, Parallelrechner mit mehreren Prozessoren können zwar – abhängig vom zu berechnenden Problem und der Güte der softwaretechnischen Zerlegung des Problems in Einzelaufgaben für die verschiedenen Kerne[95] – erheblich schneller sein als Maschinen mit einer einzigen CPU. Das ändert aber nichts an der grundlegenden Tatsache, dass auch für Parallelrechner die Berechenbarkeit im Sinne der Turingmaschine die unüberbietbare Grenze bildet: „Künstliche Intelligenzen von heute laufen schneller, paralleler, nur nicht prinzipiell anders."[96]

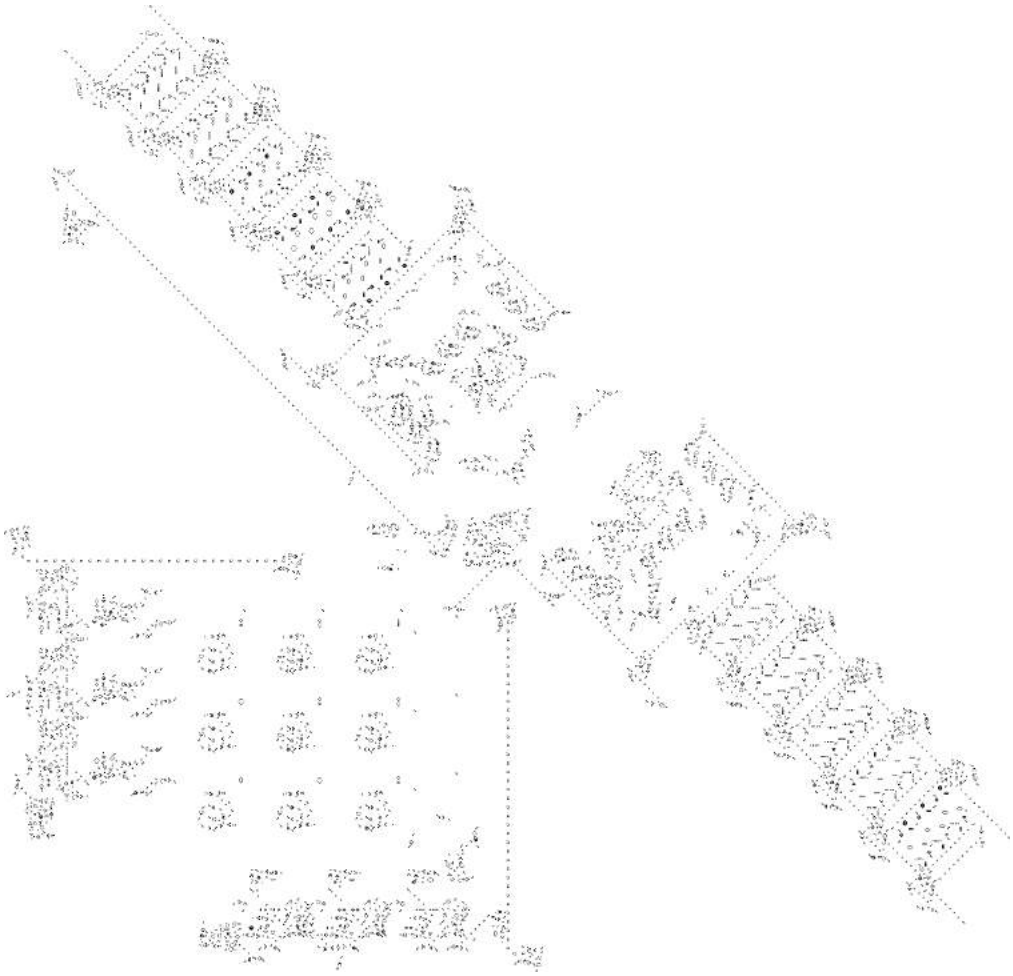
Auch ungewöhnlichere Architekturen, wie sie beispielsweise mittels universaler zellulärer Automaten umgesetzt wurden, denen in den 80er Jahren gewisse Vorzüge bei der Simulation physikalischer Prozesse zugeschrieben wurden,[97] bleiben strikt in dem von der Turingmaschine gesetzten Rahmen:

---

[92] Kittler 2000, 262.
[93] Vgl. Kuck 1989, 230.
[94] Vgl. Flynn 2004 (Internetquelle).
[95] Tatsächlich krankten parallele Systeme lange Zeit am Fehlen einer ausreichend effizienten Software für eine geeignete Aufteilung des Problems in Einzelschritte für die verschiedenen Prozessoren. Vgl. die verschiedenen in Ritter 1989 diskutierten Ansätze.
[96] Kittler 1990, 243.
[97] Vgl. Margolus/Toffoli 1987a und 1987b.



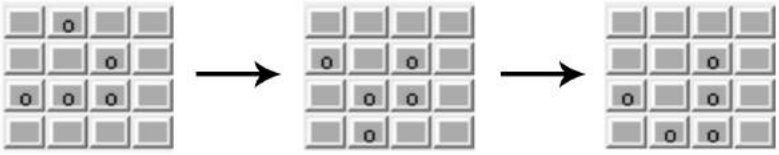
Eindimensionaler elementarer Zellularautomat mit der Regel 110. Jede neue Zeile von oben nach unten entspricht einer Generation in der Zeit. Gut zu erkennen die Spuren der Gleiter, die als Signale verwendet werden können, um logische Gatter zu erzeugen.[98]



Aufbau einer funktionsfähigen Turingmaschine im zweidimensionalen Game of Life.[99]

[98] Erstellt mit dem CELLULAR SOUNDS PROJECT. Mehr Beispiele unter Nückel 2016 (Internetquelle).
[99] Abbildung von ‚In Vollen Zügen' 2011 (Internetquelle).

Solche universalen zellulären Automaten, wie der eindimensionale elementare Automat mit der Regel 110 oder das bekanntere Game of Life von John H. Conway, nutzen dabei bestimmte periodisch oszillierende und sich über das Gitter des Automaten fortbewegende Gebilde, die sogenannten Gleiter, als Signale (vgl. hierzu die erste Abbildung auf der vorhergehenden Seite). Da diese Gleiter sich bei der Kollision untereinander oder mit anderen Strukturen auslöschen, können sie verwendet werden, um die elementaren logischen Gatter, wie sie auch in den Schaltkreisen einer CPU notwendig vorhanden sind, zu implementieren.



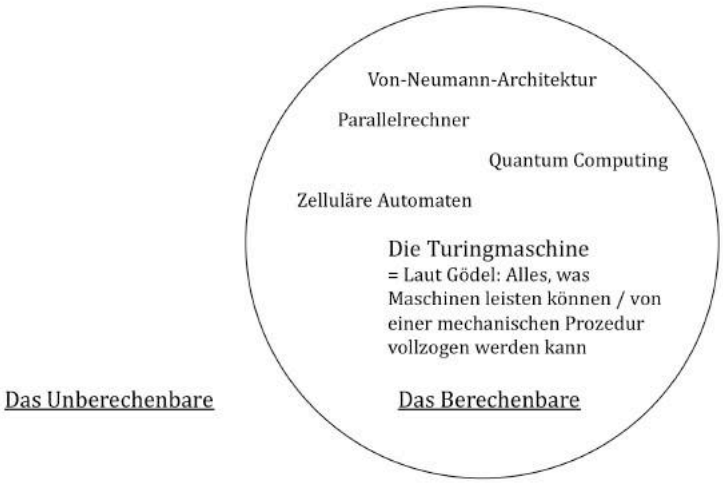
Fortbewegung eines Gleiters im zweidimensionalen Game of Life.

Dass auch universale zelluläre Automaten damit alles ausführen können, was mit Computern der von Neumann-Architektur nur irgend möglich ist, zeigt sich am augenfälligsten daran, dass sich in Game of Life gar eine voll funktionsfähige Turingmaschine konstruieren und bei der Berechnung beobachten lässt (vgl. die zweite Abbildung auf der vorhergehenden Seite).[100] Die Tatsache, dass nicht nur zweidimensionale Automaten – wie das Game of Life – universal sein können, sondern auch eindimensionale, extrem basale – wie der elementare Automat 110[101] – zeigt, dass Universalität im Sinne der Turingmaschine nicht an eine spezielle Umsetzung gebunden ist. Gleichzeitig führen aber auch andersartige Architekturen nicht aus dem Bereich der Berechenbarkeit hinaus. Die Grenze der Berechenbarkeit ist also keine Eigenschaft lediglich einiger spezieller Maschinen, so dass sie durch entsprechende Umbauten an der von Neumann-Architektur umgangen werden könnte. Vielmehr bildet sie – zumindest bislang – die unerreichbare Grenze für alle real umgesetzten Maschinen.[102]

Das gilt auch für alle bisher in die Nähe einer Umsetzung gelangten Quantenrechner: Zwar sollen Quantenrechner, die nach dem *standard quantum computing* mit *q-bits* rechnen, für bestimmte Probleme einen eklatanten Geschwindigkeitszuwachs erbringen. Sie würden beispielsweise viele asymmetrische kryptographische Methoden wie RSA Schachmatt setzen. Das *standard quantum computing* scheitert aber ebenfalls an der Lichtmauer für rechnende Maschinen. Das heißt, sämtliche nach diesem Modell umgesetzten Quantenrechner wären,

[100] Vgl. Rendell 2001. Für eine Einführung in das Game of Life vgl. Adamatzky 2010.
[101] Vgl. zur Universalität der Regel 110: Martínez 2004.
[102] Bzw. präziser: Eine Grenze für das, was mittels real umgesetzter Maschinen erkannt werden kann, wobei die damit verbundenen dann evtl. doch wieder anthropozentrischen Fragen an dieser Stelle ausgeklammert werden müssen.

was Berechenbarkeit betrifft, der Turingmaschine äquivalent, und könnten daher keine klassisch unberechenbaren Ergebnisse berechnen.[103] „The standard model of quantum computing does not allow the computation of any non-recursive functions, but does use a form of non-determinism to speed up the computation.“[104] Dass auch Modelle für das Quantenrechnen abseits des *standard quantum computings* entwickelt werden, wird noch zu behandeln sein (vgl. Kapitel 4.2.4). Bis hierhin aber bleibt die empirische Tatsache, dass es keine realen Maschinen gibt – und damit eben auch keine digitalen Maschinen – die über die Grenze der Berechenbarkeit hinaus arbeiten. Alles, was reale Maschinen zu leisten imstande sind, spielt sich innerhalb der Grenze der Berechenbarkeit ab, wie sie die Turingmaschine und Gödels Begriff der mechanischen Prozedur ziehen.



Alle bislang tatsächlich umsetzbaren Maschinenkonzepte fallen in den Bereich des Berechenbaren.

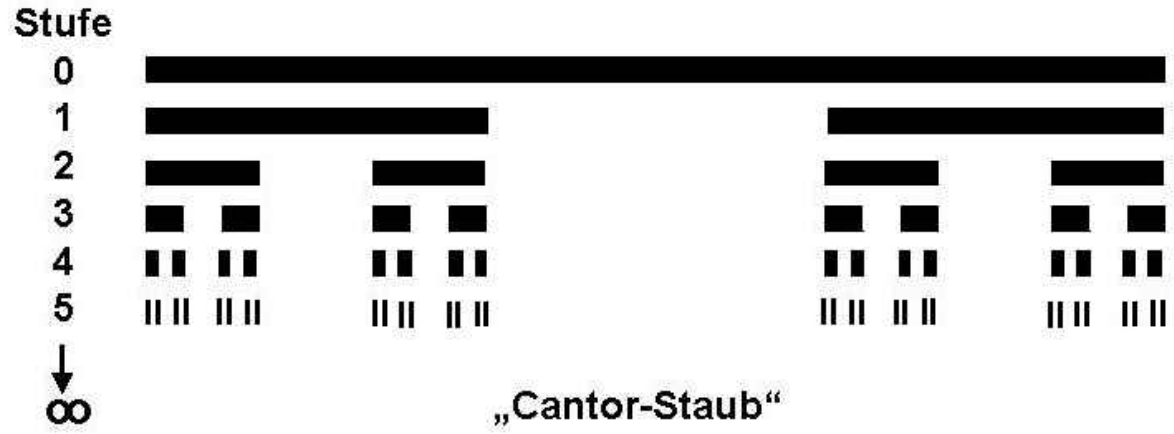
Während Supercomputer dazu dienen, Probleme schneller zu lösen, führt *Hypercomputation* aus dem Bereich der Berechenbarkeit hinaus ins Unberechenbare. Hier wird nicht *dasselbe* wie zuvor, nur schneller abgewickelt, sondern eine Lösung für Aufgaben gesucht, deren Berechnung für Turingmaschinen unmöglich ist. Es geht also nicht um Fragen der Geschwindigkeit[105] oder der Effizienz von Algorithmen, sondern um die Frage, ob bestimmte unberechenbare Tasks von Medien im weitesten Sinne nicht doch gelöst werden können. Bevor konkrete Modelle für Hypercomputation betrachtet werden, soll zunächst eine Aufarbeitung des Begriffs des *Super-Tasks* zeigen, dass die Suche nach Hypercomputern schon auf der logischen Ebene keineswegs trivial ist. Hierfür werden die Ansätze von Thomson und Benacerraf näher beleuchtet und die Frage beantwortet, ob Super-Tasks überhaupt logisch möglich sind.

[103] Vgl. dazu auch Hodges 2006, 5.
[104] Ord 2002, 31.
[105] Zumindest nicht um Fragen bezüglich endlicher Geschwindigkeiten.

4.1 Super-Tasks

4.1.1 Die logische Unmöglichkeit von Super-Tasks

Die Überlegungen zu Super-Tasks reichen weit zurück. So lässt sich Zenons Paradoxon vom Wettlauf der Schildkröte mit Achilles als einer der ersten Gedankengänge zu diesem Topos auffassen.[106] Der Begriff selbst wird aber erst 1954 durch Thomson vorgebracht. Zahlreiche Autoren aus dem Bereich der Hypercomputation beziehen sich bis heute auf seine Definition. Dies ist umso bemerkenswerter, als Thomson den Begriff des Super-Tasks nur einführt, um ihn durch die Konstruktion verschiedener Paradoxa als logisch unmöglich zu erweisen und gleich wieder zu kassieren. Das bedeutet, dass es Hypercomputation für Thomson – zumindest angesichts des Texts von 1954 – schon rein logisch nicht hätte geben können. Umso mehr lohnt es sich, seine zum Super-Task entwickelten Gedanken näher zu betrachten. Er schreibt: „Let us say, for brevity, that a man who has completed all of an infinite number of tasks (of some given kind) has completed a super-task (of some associated kind).“[107] Ein Super-Task besteht also aus unendlich vielen endlichen Tasks. Was Thomson nun widerlegen will, ist nicht, dass es Tasks gibt, die unendlich oft wiederholt werden *könnten* – denn das ist logisch denkbar – sondern die Vorstellung eines tatsächlich *ausführbaren* Super-Tasks: „But to say that some operation can be performed infinitely often is not to say that a super-operation can be performed.“[108] Dies expliziert Thomson an einem Stück Schokolade, von dem er annimmt, es könne, ähnlich dem Cantor-Staub, unendlich oft geteilt werden.



Der Cantor-Staub: Bei jedem Schritt wird jede zusammenhängende Strecke gedrittelt und ihr jeweiliges Mittelstück entfernt. Wird dieser Vorgang als Super-Task unendlich oft wiederholt, entstehen unendlich viele und unendlich kleine Streckenteile, der sogenannte Cantor-Staub.[109]

[106] Vgl. zu diesem Paradoxon die amüsante Fassung in Hofstadter 1979, 33-36.
[107] Thomson 1954, 2.
[108] Ebd.
[109] Abbildung von Klug 2010 (Internetquelle).

Anhand dieser unendlich halbierbaren Schokolade, die hier nicht im Detail verhandelt werden soll, zieht Thomson noch einmal den schon genannten Schluss: „And if something is infinitely divisible, then the operation of halving it or halving some part of it can be performed infinitely often. This is not to say that the operation *can have been* performed infinitely often.“[110] In Frage steht für Thomson also nicht, ob die Schokolade unendlich oft halbiert werden *könnte*, sondern, ob ein Mensch oder eine Maschine tatsächlich damit fertig werden kann, diese Schokolade unendlich oft zu halbieren, mithin also, ob ein Super-Task *tatsächlich abgeschlossen* werden kann:

> When we say that you can perform an infinite number of operations we are not saying that there is some one ("infinite") operation you can do, but that the set of operations ("finite" operations) which lie within your power is an infinite set. Roughly speaking: to speak of an infinity of possibilities is not to speak of the possibility of infinity.[111]

Wie aber könnte es überhaupt denkbar sein, eine unendliche Folge von endlichen Tasks zu beenden? Ein verbreitetes Modell für eine Maschine oder einen Menschen, um einen Super-Task zu erledigen, ist die sogenannte Russell-Blake-Weyl Formel.[112] Auch für diese Formel stand explizit das Paradoxon von Zenon, der Wettlauf zwischen Achilles und der Schildkröte, Pate. Sie ist die Formalisierung der Idee – die Russell schon 1914 formulierte – Prozesse mit jedem weiteren Schritt doppelt so schnell ablaufen zu lassen.

$$\tfrac{1}{2} + \tfrac{1}{4} + \tfrac{1}{8} + \ldots + \tfrac{1}{2}^{n} + \tfrac{1}{2}^{n+1} + \ldots \qquad (1)$$

Die Russell-Blake-Weyl Formel.

Der erste Task bzw. der erste Schritt läuft laut dieser Formel also in einer halben Minute ab (abhängig von der gewählten Zeiteinheit). Der zweite benötigt nur noch 15 Sekunden, der dritte 7,5 Sekunden, usw. Das bedeutet, dass der aus unendlich vielen Einzelschritten bestehende Gesamtprozess des Super-Tasks abgelaufen sein wird, bevor der Grenzwert der oben angegebenen Folge erreicht wird. D. h. ab Beginn der zweiten Minute wird der nach der Russell-Blake-Weyl Formel ablaufende Super-Task abgeschlossen sein. Wie Russell nun schreibt, ist es auf diese Weise logisch denkbar, die reelle Zahl π in ihrer gesamten Ausdehnung anzuschreiben.[113] Die Erledigung eines solchen beschleunigenden Super-Tasks wäre nämlich zwar „*medically* impossible“[114], logisch aber sehr wohl möglich.

---

[110] Thomson 1954, 2.
[111] Ebd., 3.
[112] Vgl. Russell 1915, 177; Blake 1926, 650f.; Weyl 1927, 66; Russell 1936, 143f. Vgl. auch Copeland 2002, 282f. und 289.
[113] Vgl. Russell 1936, 143.
[114] Ebd. [sic] Vgl. auch Thomson 1954, 4.

Auch wenn weder Blake noch Weyl bei Thomson explizit Erwähnung finden, bezieht er sich dennoch auch auf die in dieser Formel ausgedrückte Idee des jeweils doppelt so schnell ausgeführten nächsten Schritts. So gibt er eine abgeleitete Formel an, mit der sich die Dauer eines einzelnen Schrittes *n* angeben lässt, statt die Summe aller bisherigen Schritte:

$$\frac{1}{2^{n-1}} \qquad (2)$$

Dies ließe sich unter Einbeziehung tatsächlicher Zeitangaben so erweitern:

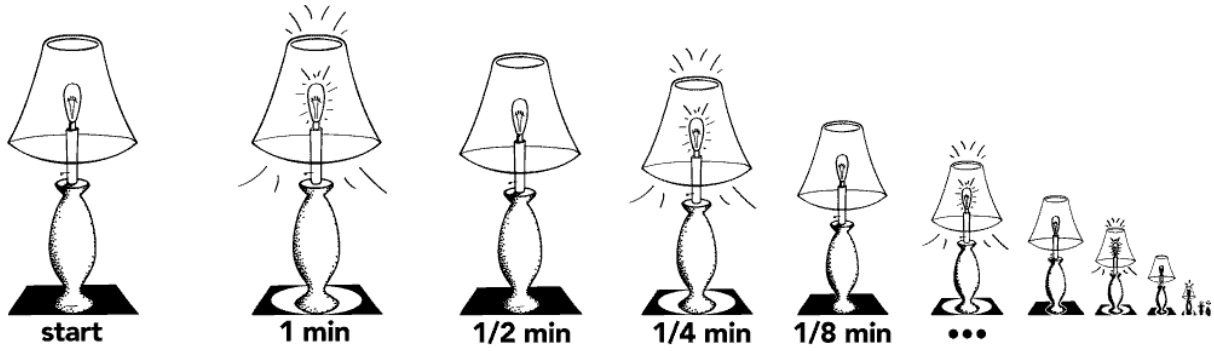
$$\Delta t_n = \Delta t_1 * \frac{1}{2^{n-1}} \qquad (3)$$

wobei Δt_1 die Dauer des ersten Operationsschrittes wäre und *n* der Schritt, dessen Dauer Δt_n auszurechnen ist.

Auf Basis dieser Formel und der Idee der jeweils nur halb so langen Operationsschritte will Thomson zeigen, dass es einen Super-Task logisch nicht geben kann. Das Beispiel, das er für diesen Zweck heranzieht, und das im Bereich der Theorie der Hypercomputation in noch zu zeigender Weise Schule gemacht hat, ist das der Leselampe, auch *Thomson's lamp* genannt.
Diese Lampe verfügt über einen Knopf, dessen Betätigung die Lampe anschaltet, wenn sie zuvor aus war, und ausschaltet, wenn sie zuvor an war. Wird dieser Knopf also wiederholt betätigt, so wird die Lampe entsprechend oft zwischen den Zuständen AN und AUS wechseln. Dabei gilt auch: Wenn die Lampe zu Beginn einer solchen Serie von Knopfbetätigungen aus ist, ihr Knopf *n*-mal betätigt wird und *n* eine gerade Zahl ist, dann ist die Lampe nach Durchführung dieses Vorgangs erneut aus. Dieser Zusammenhang lässt sich selbstredend auch für eine ungerade Anzahl von Betätigungen in entsprechender Weise herstellen und ebenso für den Fall, dass die Lampe zu Beginn an war. Das bedeutet: Bei einer endlichen Anzahl von Tasks, in diesem Fall also das Betätigen des Knopfes der Leselampe, lässt sich – solange der Anfangszustand der Lampe bekannt ist und auch, ob der Schalter eine geradzahlige Anzahl oder eine ungeradzahlige Anzahl betätigt wird – stets sagen, welchen Zustand die Lampe nach Beendigung des Vorgangs haben wird, ob sie also an oder aus sein wird. Wie aber steht es um die Möglichkeit einer solchen Voraussage, wenn der Knopf der Lampe unendlich oft betätigt wird, wenn also aus den jeweils für sich endlichen Tasks, die nur daraus bestehen, den Knopf zu drücken, ein Super-Task gebildet wird?[115]

[115] Vgl. Thomson 1954, 5.

Bei der Aufstellung dieses Gedankenexperiments greift Thomson auf seine Version der Russell-Blake-Weyl Formel zurück, setzt allerdings eine von der in der oben angegebenen Formel (1) verschiedene Dauer Δt_1 für den ersten Operationsschritt, statt einer halben Minute wie in (1) eine ganze Minute. Entsprechend bleibt die Leselampe nach der ersten Betätigung des Schalters eine Minute lang aus, wird dann erneut betätigt, worauf sie nur die Hälfte dieser Zeit, nämlich ½ Minute an bleibt, wird dann für ¼ Minute ausgeschaltet usw.[116]



Thomson's lamp.[117]

Dieser Super-Tasks wird damit nach dem Vorbild der Formel (1) abgeschlossen sein, bevor die zweite Minute beginnt. In welchem Zustand aber ist dann die Leselampe? Sie kann nicht an sein, denn, da der Super-Task aus unendlich vielen Tasks besteht, wurde sie jedes mal, wenn sie angeschaltet wurde, auch unweigerlich wieder ausgeschaltet. Sie kann aber auch nicht aus sein, da sie nach jedem Ausschalten stets wieder angeschaltet wurde. Da die Lampe also, wie Thomson schreibt, in keinem dieser Zustände sein kann, sie aber nun einmal definitionsgemäß an oder aus sein muss, zeigt sich in der Konzeption des Super-Tasks ein Widerspruch, der dieses Konzept ad absurdum führt. Die Leselampe unendlich oft zu betätigen und diesen Super-Task dabei tatsächlich abzuschließen ist deswegen für Thomson nicht nur „*medically* impossible", wie Russell meint, sondern auch logisch unmöglich.[118] Daher ist es laut Thomson ebenso logisch unmöglich, eine Maschine zu bauen, die, wie von Russell gedacht, mittels desselben beschleunigenden Verhaltens in knapp zwei Minuten den gesamten Dezimalausdruck von π niederschreibt.

[116] Thomson gibt keine Angabe, ob die Lampe im ersten Operationsschritt an oder aus sein soll. Da dies auf die folgende Argumentation keinerlei Einfluss hat, wurde hier der zur Abbildung aus der Stanford Encyclopedia of Philosophy passende Anfangszustand gewählt. Auch diese Abbildung ließe sich allerdings genau andersherum deuten.
[117] Abbildung von Manchak/Roberts 2016 (Internetquelle).
[118] Thomson 1954, 5.

Thomson belegt dies, indem er das Alternieren der Leselampe in einen anderen Kontext setzt: Er kombiniert die Russellsche Maschine mit einer zweiten, die jeweils ‚0' oder ‚1' anzeigt, je nachdem, ob die von der ersten Maschine gerade berechnete Stelle gerade oder ungerade ist. Für die so erweiterte Russellsche Maschine ergibt sich dann die selbe Kontradiktion wie im Falle der unendlich oft betätigten Leselampe. In Thomsons eigenen Worten:

> For if such a machine [Russells Maschine] is (logically) possible so presumably is one that records the parity, 0 or 1, of the integers written down by the original machine as it produces them. Suppose the parity-machine has a dial on which either 0 or 1 appears. Then, what appears on the dial after the first machine has run through all the integers in the expansion of π?[119]

Thomsons Argumentation greift im Folgenden weiter als diese ersten Beispiele. Der grundlegende Gedanke jedoch bleibt derselbe: Durch das Aufweisen eines Widerspruchs in der Konzeption von Super-Tasks wird gezeigt, dass der tatsächliche Abschluss einer unendlichen Reihe von Tasks logisch unmöglich ist, für Menschen, wie für Maschinen. In gewissem Sinne scheint es Thomson dabei auch um die Differenz von mathematischer und – wenn man so will – realer Unendlichkeit zu gehen. So konvergiert beispielsweise die Reihe, die dem Beispiel der Leselampe zugrunde lag, gegen den Wert 2. Wer die oben genannte Formel aber tatsächlich ausführt, wird niemals bei 2 ankommen, da es keine letzte Addition gibt, nach der die Berechnung beendet wäre.[120] Diese Differenz schlägt sich auch in den mathematischen Methoden selbst nieder: So gibt es, weil das mathematische Arbeiten mit Grenzwerten eine andere – „*demonstrably different*"[121] – Methode ist als die des Berechnens der Summen von finiten Termen, auch keine der letzteren entsprechende Methode für das Berechnen der Summe einer infiniten Sequenz.[122]

Es bleibt bezeichnend, dass Thomson den Begriff des Super-Tasks nur definiert, um ihn als logisch unmöglichen Begriff zu erweisen. Zwar vertritt er nicht durchgängig diese Meinung, sondern spricht auch davon, dass der Begriff noch nicht genügend untersucht sei.[123] Das Reden über Super-Tasks aber hält er, da ihm die Konstruktion eines logischen Widerspruchs für die Annahme eines tatsächlich abschließbaren Super-Tasks gelungen scheint, in einer an den frühen Wittgenstein erinnernden Geste für sinnlos: „And if I am correct in supposing that talk of super-tasks is *senseless*, then this kind of talk cannot *give* a sense to anything."[124]

---

[119] Ebd.
[120] Vgl. ebd., 7f. und 9.
[121] Ebd., 9.
[122] Vgl. ebd.
[123] Vgl. ebd., 6 und Benacerraf 1962, 767n3.
[124] Thomson 1954, 9.

4.1.2 Widerlegung der logischen Unmöglichkeit von Super-Tasks

Umfassenden Bezug auf die Arbeit Thomsons und seine Definition des Super-Tasks nimmt Benacerraf in einem Text von 1962. Dieser spricht dort von Thomsons Arbeit als einem „remarkable and stimulating paper“[125] und geht explizit auf *Thomson's lamp*, das Beispiel der Leselampe ein. Er schreibt dazu: „Rarely are we presented with an argument so neat and convincing. This one has only one flaw. It is invalid.“[126]

Thomsons Argument nämlich, die Lampe, die ab dem Zeitpunkt t_0 unendlich oft ein- und ausgeschaltet wurde, könne zum Zeitpunkt t_1 – das ist laut Definition Benacerrafs der erste Zeitpunkt nach Ablauf des vollständigen Super-Tasks – weder an noch aus sein, enthält für Benacerraf den folgenden schwerwiegenden Fehler: Zwar sind die von Thomson dargelegten Gedanken im Grunde richtig; sie gelten aber nur für Zeitpunkte *vor* t_1!

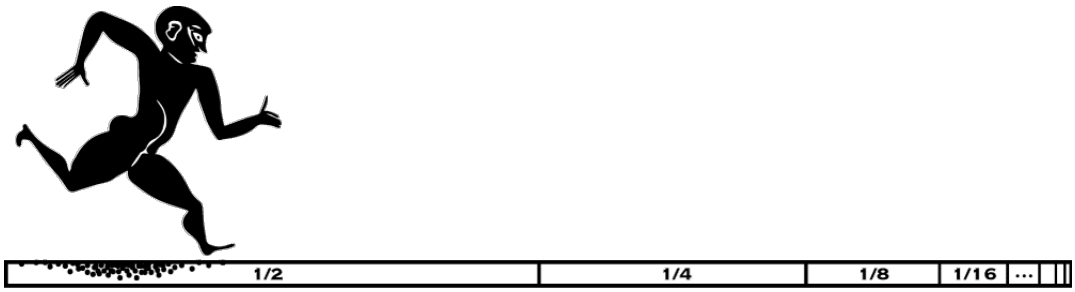
> Nothing whatever has been said about the lamp at t_1 *or later*. […] The only reasons Thomson gives for supposing that *his* lamp will not be off [or on] at t_1 are ones which hold only for times *before* t_1. The explanation is quite simply that Thomson's instructions do not cover the state of the lamp at t_1, although they do tell us what will be its state at every instant *between* t_0 and t_1 (including t_0).[127]

Was Benacerraf Thomson damit vorwirft, ist, jener habe im Verlauf seiner Argumentation den Rahmen des von ihm selbst definierten Super-Tasks gesprengt und stattdessen fälschlicherweise begonnen, nach den Eigenschaften eines noch größeren sogenannten Super-Duper-Tasks zu fragen: „If a super-task is a task sequence of order type ω, then a *super-duper task* [sic, hier einmalig bei B. ohne Bindestrich zwischen ‚duper' und ‚task'] is the result of tacking an extra (ωth) task at the end of a super-task.“[128]

Um diese Behauptung zu beweisen, greift Benacerraf direkt auf Zenons Paradoxon vom Rennkurs zurück, nur dass er nicht Achilles und die Schildkröte zum Wettlauf antreten lässt, sondern einen einzelnen Djin („*genie*“ im Original). Dieser Djin läuft die nach der Russell-Blake-Weyl Formel unterteile Rennstrecke ab, so dass er zuerst die Hälfte des Rennkurses, dann ein weiteres Viertel, dann ein weiteres Achtel, usw. hinter sich bringt. Benacerraf geht es nicht um die alte Frage Zenons, ob Bewegung angesichts der Tatsache, dass der Läufer hier unendlich viele Teilstrecken zurücklegen muss, denn überhaupt möglich sein kann. Die Absicht hinter seiner Neubesetzung des Rennkurses ist es vielmehr, ein Beispiel zu finden, für das die Frage Thomsons, welchen Zustand das ausführende Objekt des Super-Tasks nach

[125] Benacerraf 1962, 765.
[126] Ebd., 768.
[127] Ebd. Benacerraf bezieht sich mit „*his* lamp“ auf einen fiktiven Benutzer namens Bernard, dessen Lampe nach Ablauf des Super-Tasks aus sein wird, während die Lampe seines Pendants Aladin an ist.
[128] Ebd., 772.

dessen Abschluss haben soll, keinen logischen Widerspruch erzeugt. Die angesichts von Thomsons Paradoxon interessante Frage ist also: Ist es denkbar, dass der Djin alle Punkte des Rennkurses abläuft und dann verschwindet? Ist dies nämlich möglich, dann muss in diesem Fall die Frage, wo der Djin nach dem Durchlaufen aller Punkte der Rennstrecke ist, mit *Nirgends* beantwortet werden. Gelingt es dem Djin daher, nach Abschluss seines Super-Tasks auf logisch konsistente Weise zu verschwinden, so macht das jede Frage nach seinem Zustand unnötig und ein Gegenbeispiel für Thomsons Paradoxon ist gefunden.

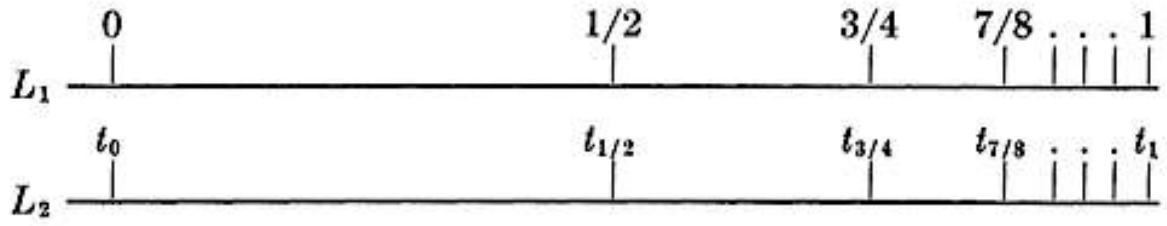


Wie der Läufer in Zenons Paradoxon muss auch der Djin unendlich viele Teilstrecken hinter sich bringen.[129]

Kann der Djin auf logisch konsistente Weise verschwinden und dabei den gesamten Super-Task abschließen? Wenn er verschwindet, muss es einen Punkt geben, von dem er verschwindet. Dieser Punkt wird als der Punkt *p* definiert. Entweder liegt *p* nun zwischen zwei Punkten auf der Rennstrecke, oder rechts von all diesen, d. h. auf oder nach Punkt 1 (vgl. die Abbildung auf der folgenden Seite). Liegt *p* zwischen zwei Punkten auf der Rennstrecke, dann hat der Djin nicht alle Punkte abgelaufen, bevor er verschwindet, und der Super-Task wurde nicht komplett ausgeführt. Hat er aber den Punkt 1 rechts von allen Punkten der Rennstrecke erreicht, dann bedeutet dies für den Super-Task, dass er zwar vollständig ausgeführt werden konnte, dass aber zugleich ein über ihn hinausgehender Super-Duper-Task im Sinne Benacerrafs vollbracht werden musste. Schließlich hat der Djin so nicht nur alle Punkte der Rennstrecke abgelaufen, sondern er musste, um dies zu tun, notwendig noch einen weiteren Punkt ablaufen. Damit aber wäre Thomsons Argumentation rechtmäßig. Immer noch wäre zwar korrekt, dass Thomson, ohne es explizit zu machen, den Rahmen des von ihm definierten Super-Tasks verlässt. Wenn die Erledigung eines Super-Tasks aber stets zusätzliche Schritte erfordert, d. h. einen Super-Duper-Task notwendig macht, dann ist es legitim für Thomson, diese zusätzlichen Schritte implizit mit einzubeziehen.[130]

[129] Abbildung von Manchak/Roberts 2016 (Internetquelle).
[130] Vgl. Benacerraf 1962, 774f.

Benacerraf zeigt nun, dass auch unter diesem erweiterten Gesichtspunkt das Paradoxon von Thomson nicht haltbar ist. Dazu greift er auf die bereits angesprochene Darstellung zurück, in der die räumlich und zeitlich nach der Russell-Blake-Weyl Formel unterteilte Rennstrecke durch die Linien L_1 und L_2 repräsentiert wird:



Schematische Darstellung der von Zenon entlehnten Unterteilung der Rennstrecke nach der Russell-Blake-Weyl Formel. L_1 repräsentiert die spatiale Unterteilung, L_2 die temporale.[131]

Wie Benacerraf argumentiert, muss der Djin zwar entweder von einem bestimmten Punkt *auf* der Rennstrecke – d. h. von einem Punkt im Intervall [0, 1[– oder von einem bestimmten Punkt *nach* der Rennstrecke – d. h. von einem Punkt im Intervall [1, ∞] – verschwinden. Die entscheidende Frage ist aber, ob der Djin, wenn er von einem bestimmten Punkt verschwindet, diesen Punkt damit auch besetzt hat („occupied“[132] im englischen Original):

> 'He disappeared at 1' could mean either that 1 is the last point he occupied *or* that 1 is the first point he didn't occupy, just as to have disappeared at t_1 could involve *either* that t_1 was his last moment on earth *or* that t_1 was earth's first moment without him.[133]

In diesem Sinne lassen sich die markierten Punkte auf den Linien L_1 und L_2 auf zwei unterschiedliche Weisen betrachten: Jeder Punkt auf einer der beiden Linien (wie z. B. der Punkt bei ½ bzw. $t_{½}$) teilt diese Linie in zwei Mengen von Punkten auf, in die, die links von ihm liegen, und die, die rechts von ihm liegen. Zu welcher der beiden Mengen aber der teilende Punkt selbst gehört, bleibt der Wahl des Betrachters überlassen. Es gibt keine verpflichtende Regelung, ihn der linken oder der rechten Punktmenge zuzuweisen.[134] Ob der Djin, der nach Zurücklegen der gesamten Bahn vom Punkt 1 verschwindet, diesen Punkt 1 auch *besetzt*, ist daher nicht definitiv zu beantworten, sondern hängt von der arbiträren Entscheidung ab, zu welcher Menge dieser Punkt gezählt werden soll; zur Menge der Punkte, die der Djin besetzte, oder zur Menge der Punkte, die der Djin nicht besetzte.[135]

---

[131] Ebd., 775.
[132] Ebd.
[133] Ebd.
[134] Das entspricht den Notationen [0, ½] und]½, 1] für eine Zuweisung des teilenden Punktes ½ zur linken Punktmenge, oder entsprechend [0, ½[und [½, 1] für eine Zuweisung zur rechten Punktmenge.
[135] Vgl. ebd., 775f.

Um dieses bislang recht abstrakte Argument zu verdeutlichen, gibt Benacerraf ein Beispiel, das E. B. Davies Idee der *Shrinking Machines* vorwegzunehmen scheint (vgl. Kapitel 4.2.3). Er nimmt schlicht an, dass der Djin proportional zum noch abzulaufenden Weg schrumpft. Am Punkt 0 hat er demnach seine volle Größe, am Punkt ½ nur noch die Hälfte, usw.:

> Clearly, now, he occupied every point to the left of 1 (I can tell you exactly when and how tall he was at that point), but he did not occupy 1 (if he followed instructions [sic], there was nothing left of him at 1). Of course, if we *must* say that he vanished at a point, it must be at 1 that we must say that he vanished, but in this case, there is no temptation whatever to say that he occupied 1. He couldn't have. There wasn't enough left of him.[136]

Dieses Beispiel des schrumpfenden Djins zeigt, dass es tatsächlich logisch konsistente Bedingungen gibt, unter denen der Djin alle Punkte der Rennbahn abläuft, den Punkt 1 aber nie besetzt. Das bedeutet nicht, dass der Djin schrumpfen *muss*, um zu verhindern, dass er Punkt 1 besetzt. Ob er diesen besetzt, ist, wie oben dargestellt, eine Frage der Sichtweise, d. h. abhängig von der arbiträren Entscheidung, zu welcher Punktmenge Punkt 1 gezählt werden soll.

Benacerrafs Neubesetzung des Rennkurses beweist damit, dass es Möglichkeiten gibt, den Abschluss von Super-Tasks logisch konsistent zu denken, ohne dass ihre Ausführung zusätzliche, über den Super-Task hinausgehende Schritte notwendig machen würde. Auf ebensolche zusätzlichen Schritte nimmt Thomsons Paradoxon von der Leselampe aber implizit Bezug, wenn nach dem Zustand der Lampe nach Abschluss des Super-Tasks gefragt wird. Da aber ein Super-Task logisch denkbar ist, der abgeschlossen werden kann, ohne dass daraus etwas über den Zustand des ausführenden Systems *nach* dem Super-Task gesagt werden muss, ist der von Thomson konstruierte Widerspruch nicht haltbar.[137] Für das Alltagsverständnis mag es notwendig erscheinen, eine Verbindung zwischen dem letzten Akt des Super-Tasks und dem darauffolgenden nächsten Zustand des den Super-Task ausführenden Systems zu unterstellen. Dies ist aber keineswegs logisch notwendig:

> It has not been shown that the existential compulsion we feel which drags us from one moment to the next is a *logical* one – but *this too* must be shown in proving that to occupy all the Z-points [die Punkte auf der Rennbahn vor dem Punkt 1], the runner must occupy a point outside of Z.[138]

Eine Sichtweise, die sowohl Foucaults Betonung der Brüche und des Diskontinuierlichen als auch Heideggers Zeitlichkeit auf den Plan zu rufen scheint.[139]

---

[136] Ebd., 776f.
[137] Vgl. ebd., 777.
[138] Ebd., 778.
[139] In einem entfernt an sokratische Reden erinnernden Dialog geht Benacerraf noch einmal explizit auf die Frage ein, ob die fehlende Prämisse, dass der letzte Schritt des Super-Tasks den darauffolgenden ersten nicht

Im Unendlichen mögen zudem andere Zusammenhänge herrschen als im Endlichen. Die Frage, was es bedeutet, eine Handlung abzuschließen, muss hier jeweils verschieden beantwortet werden:

> On the one hand "complete" can refer to the execution of a final action. This sense of completion does not occur in Zeno's Dichotomy, since for every step in the task there is another step that happens later. On the other hand, "complete" can refer to carrying out every step in the task, which certainly does occur in Zeno's Dichotomy. [O]ne can see that the Zeno Dichotomy cannot be completed in the first sense. But it can be completed in the second. The two meanings for the word "complete" happen to be equivalent for finite tasks, where most of our intuitions about tasks are developed. But they are not equivalent when it comes to supertasks.[140]

Bemerkenswerterweise ist Benacerraf nicht der Ansicht, er hätte gezeigt, dass Super-Tasks logisch möglich sind: „I think it should be clear that, just as Thomson did not establish the impossibility of super-tasks by destroying the arguments of their defenders, I did not establish their *possibility* by destroying his (supposing that I did destroy them)."[141] Auch wenn er die nach eigenen Worten empirische Vermutung äußert, dass es sich bei dem Begriff des Super-Tasks um die Kreuzung zweier Begriffe handelt, die so keinen Sinn ergeben,[142] bleibt er im abschließenden Urteil vorsichtig: „Possibly someday someone will find such a description [of a completed infinite sequence of tasks]."[143]

Benacerraf zeigt also die Inkorrektheit von Thomsons Paradoxon, eine Tatsache, der selbst Thomson später zugestimmt hat.[144] Dass es Benacerraf dabei vor allem anderen um Sprache geht, um das, was mit dieser Sprache beschreibbar ist, und nicht um tatsächlich baubare Maschinen,[145] ändert nichts daran, dass mit dem Scheitern eines der grundlegenden Beweisversuche für die logische Unmöglichkeit von Super-Tasks die Tür für weitere Überlegungen wieder offen steht, und es erst so wieder eine gewisse Berechtigung haben kann, Maschinen wie die noch folgenden Konzepte aus dem Bereich der Hypercomputation als mehr als nur ein Gedankenspiel zu verhandeln.

---

zum Super-Task gehörenden Zustand bestimmt – ob also z. B. die Lampe nach Ablauf des Super-Tasks an sein muss, wenn der letzte Schritt des Super-Tasks es war, sie anzuschalten – nicht schlicht zum Beweis hinzugefügt werden kann. Er zeigt, dass dies keine große Änderung verursacht. Vgl. ebd., 779f.

[140] Manchak/Roberts 2016 (Internetquelle).

[141] Benacerraf 1962, 778f.

[142] Vgl. ebd., 784. Benacerraf schreibt im Detail: „We have what appears to be a conceptual mismatch. Sequences of tasks do not exhibit the characteristics of sequences that lend themselves to proofs of infinity. And since there seems to be an upper bound on our ability to discriminate (intervals, say) and none on how finely we cut the task, it appears that we should never be in a position to claim that a super-task had been performed. But even if this is true, it only takes account of one kind of super-task, and, as I argue above, it hardly establishes that even this kind constitutes a logical impossibility." Ebd., 783f.

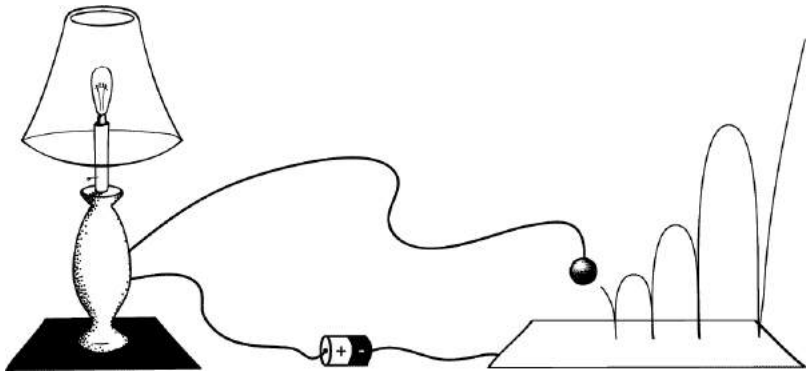
[143] Ebd., 782.

[144] Vgl. Thomson 1970, sowie Copeland 2002, 286.

[145] Vgl. Benacerraf 1962, 782n6.

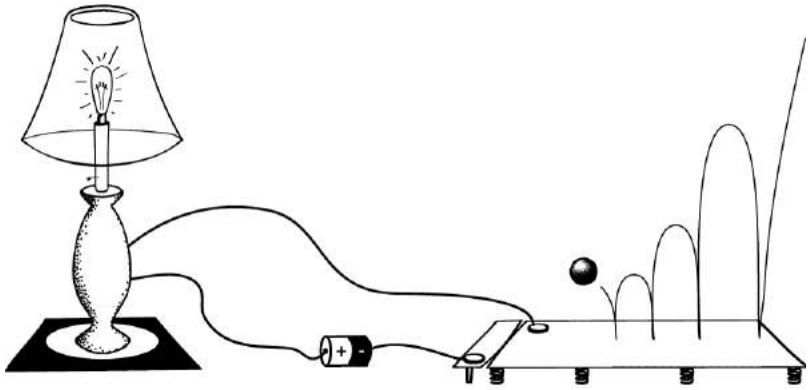
4.1.3 Eine elektromechanische Umsetzung eines Super-Tasks

Es existiert ein Modell, das, angelehnt an *Thomson's lamp*, einen Super-Task als elektromechanisches System ohne Bezug auf menschliches Schalten umsetzt. Dieses Modell von Earman und Norton (1996) besteht aus Lampe, Stromkreis mit Spannungsversorgung, Kontaktplatte und einem hüpfenden Ball. Im ersten der beiden möglichen Fälle schließt der Kontakt des Balles mit der Platte den Stromkreis und die Lampe leuchtet dann entsprechend.



Der Kontakt des Balles mit der Platte schließt den Stromkreis und schaltet die Lampe an.

Da es sich um einen idealisierten Ball handelt, der unendlich viele kleiner und schneller werdende Sprünge ausführt, bis er nach endlicher Zeit zur Ruhe auf der Platte kommt, wird die Lampe wie in Thomsons Beispiel unendlich oft an und aus geschaltet. Der Kniff ist nun, dass sich in diesem Modell klar sagen lässt, welchen Zustand die Lampe nach Erledigung des Super-Tasks haben wird: Sie wird an sein, da die Kugel auf der Platte zur Ruhe gekommen ist. Auch der umgekehrte Fall ist möglich, wenn die Parameter vertauscht werden:



Der Kontakt des Balles mit der Platte unterbricht den Stromkreis und schaltet die Lampe aus.[146]

Damit ist gezeigt, dass idealisierte, aber deswegen nicht weniger logisch mögliche Modelle existieren, die Thomsons Paradoxon von der Unmöglichkeit der Super-Tasks widerlegen.

[146] Beide Abbildungen auf dieser Seite von Manchak/Roberts 2016 (Internetquelle).

4.2 Hypercomputer

Während die an Thomsons Leselampe angelehnte elektromechanische Umsetzung lediglich ein Beispiel für die Umsetzung eines *Super-Tasks* ist, werden im Folgenden vier elaboriertere Modelle für *Hypercomputation* vorgestellt. Betrachtet werden zuerst die *Oracle-Machines*, erdacht von Alan Turing *himself*, was zeigt, dass er selbst schon an die konzeptuelle Möglichkeit von Hypercomputern *avant la lettre* gedacht hat. Mit der *Accelerating Turing Machine* wird ein Modell betrachtet, das zwar nach heutigem Kenntnisstand nicht umsetzbar ist, noch 1850 aber mit den existierenden Theorien der Newtonschen Physik weitgehend konform gegangen wäre. Noch stärker gilt dies für das dritte Beispiel, die *Shrinking Machines* von E. B. Davies.[147] Allerdings gibt es auch für ein relativistisches Universum, wie es heute als das unsere angenommen wird, Modelle für Hypercomputer. Die Hoffnungen ruhen dabei vor allem auf speziellen Versionen von *Quantenrechnern*. Als vierter Punkt werden daher Ansätze aus dem Bereich des Quantencomputing skizziert, von denen zumindest einige erhoffen, dass diese eine reale Umsetzung finden und die Ausführung von Super-Tasks tatsächlich möglich machen könnten.

4.2.1 Oracle Machines

Kittler wird nicht müde, Turings Arbeit ‚On Computable Numbers' von 1936 als dessen Dissertation zu bezeichnen.[148] Es ist nachvollziehbar, warum ihm an einer solchen Behauptung gelegen ist: Zum einen geht es ihm um die dadurch mögliche akademische Geste, nämlich darauf hinzuweisen, dass die Geburtsurkunde des speicherprogrammierbaren Computers keine allein technische Leistung ist, sondern ihrem Autor auch zu dessen Doktorwürde gereichte. Zum anderen kann Kittler so zwischen der „wohl folgenreichsten Dissertation der Mathematikgeschichte"[149] und der „folgenreichste[n] Magisterarbeit aller Zeiten"[150], nämlich Claude Elwood Shannons ‚A Symbolic Analysis of Relay and Switching Circuits' von 1938, eine Kopplung erstellen. Das ändert aber nichts daran, dass die Behauptung, ‚On Computable Numbers' wäre in irgendeiner Form eine akademische Abschlussarbeit, falsch ist.

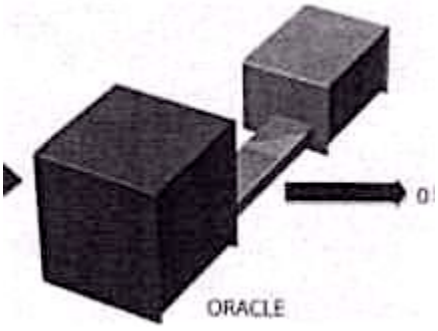
[147] Der in diesem Kontext weder mit Martin Davis, dem Autor von *Computability and Unsolvability* (1958), noch mit Donald Davies, dem Mitarbeiter Turings, der zum Ärger seines Chefs einige Programmierfehler in ‚On Computable Numbers' aufzeigte (vgl. Copeland 2004, 92), verwechselt werden darf.
[148] So in Kittler 1993, 287; 1996a, 109; 2000a, 260; 2000b, 105; 2002, 279.
[149] Kittler 2000a, 260.
[150] Kittler 1996a, 110.

‚On Computable Numbers' ist Turings zweite Veröffentlichung – die erste war ein lediglich zwei Seiten langer Beweis unter dem Titel ‚Equivalence of Left and Right Almost Periodicity' (1935).[151] Keiner dieser selbst für mathematische Abschlussarbeiten schlicht zu kurzen Texte aber ist Turings Dissertation.[152] Seine Doktorarbeit verfasste Turing vielmehr 1938 unter dem Titel ‚Systems of Logic Based on Ordinals', 1939 wurde sie veröffentlicht.

Das Bemerkenswerte ist: Während der Fokus in ‚On Computable Numbers' eher auf das gerichtet ist, was berechenbar ist, wendet er sich in der Dissertation hin zum Unberechenbaren. Ein Umstand, der Copeland und Proudfoot gar dazu bewegt, Turing als einen Pionier der realen Umsetzung von Hypercomputation auszurufen, als spräche Turing in seiner Doktorarbeit von einem tatsächlich zu bauenden Hypercomputer[153] – was Andrew Hodges mit den Worten kommentiert: „This must surely be the most ludicrous proposal ever aired in Scientific American. At first I wondered if it was an April Fool."[154]

Tatsächlich führt Turing in ‚Systems of Logic Based on Ordinals' eine Maschine ein, die unberechenbare Funktionen eben doch berechnen kann. Diese Maschine, die sogenannte *Oracle-Machine*, ist allerdings, verglichen mit der Maschine von 1936, „much less practical in its design and not intended to represent a method that could be followed by human mathematicians."[155] Daher Hodges' Kritik an dem Versuch Copelands und Proudfoots, Turing den Gedanken an eine reale Umsetzung zuzuschreiben. Davon unbetroffen bleibt, dass Turings *Oracle-Machine* oder *O-Machine* als das paradigmatische Modell für alle späteren Hypercomputer gilt.[156]

Oracle-Machines haben ihren Namen daher, dass sie aus einer gewöhnlichen Turingmaschine bestehen, die allerdings um eine entscheidende Komponente erweitert wurde, nämlich um ein



Orakel.[157] Wie dieses Orakel intern funktioniert, bleibt offen. Turing macht keinerlei Angaben, wie das Innere dieser Black Box beschaffen sein soll. Nun geht es darum, dass dieses Orakel per Definition in der Lage ist, auf bestimmte Anfragen der angeschlossenen Turingmaschine die richtige Antwort zu geben. So kann diese Turingmaschine, wenn sie auf Stellen stößt, die nicht von ihr selbst berechnet werden können,

[151] Vgl. Copeland 2004, 6. Turings Arbeit von 1935 ‚On the Gaussian Error Function' ist nicht veröffentlicht.
[152] Vgl. dazu Alexander Kluges irritierte (oder wissende) Nachfrage in Kittler 2002, 279: „Nicht länger?"
[153] Vgl. Copeland/Proudfoot 1999, 99. Vgl. auch Copeland/Proudfoot 2000.
[154] Hodges (Internetquelle).
[155] Ord 2002, 14.
[156] Vgl. ebd., 16.
[157] Die Abbildung im Text zeigt einen Ausschnitt der Graphik von Copeland/Proudfoot 1999, Abbildung in dieser Form von Hodges (Internetquelle) und vom Verf. nachbearbeitet.

das angeschlossene Orakel konsultieren und erhält von dort die fehlenden Antworten. Dafür verfügen diese *O-Maschinen* über eine gewöhnliche Tabelle. In dieser Tabelle werden lediglich einige Stellen offen gelassen: Tritt dann eine zu diesen offenen Stellen gehörige Kombination aus Zustand und Symbol ein, so findet die Maschine in der Tabelle nicht das Symbol für den nächsten Zustand, in den sie wechseln soll, sondern ein Symbol, das sie dazu bringt, eine Anfrage an das Orakel zu stellen. Je nach Art der *O-Machine* kann diese Anfrage das *Satisfactoriness Problem* betreffen, das Halteproblem, oder irgendeine andere unberechenbare Aufgabe. Die Turingmaschine wird die Antwort des Orakels abwarten, dann in einen von dieser Antwort eindeutig bestimmten Zustand wechseln und mit der Berechnung fortfahren, bis sie fertig ist oder das Orakel erneut konsultieren muss.[158]

Interessanterweise sind dabei verschieden starke *O-Machines* möglich. Die schwächste Variante ist die *O-Machine* mit einem Orakel Ø, d. h. einem leeren Orakel, das keine Antworten geben kann. Im Prinzip hat diese *O-Machine* damit keinen Zugang zu einem Orakel und kann daher nur das berechnen, was im Sinne der Turingmaschine berechenbar ist. Sie ist äquivalent zu einer gewöhnlichen Turingmaschine. Wie stark eine *O-Machine* ist, hängt von der Art des angeschlossenen Orakels ab. Es gibt dabei keine *O-Machine*, die alle Probleme lösen könnte, so dass der Inhalt des jeweiligen Orakels bestimmt, was die spezielle *O-Machine* berechnen kann. Dies zeigt sich auch daran, dass keine *O-Machine* in der Lage ist, das zu ihr gehörige Halteproblem zu lösen. Turings Beweis aus ‚On Computable Numbers' bleibt im Prinzip auch hier gültig. Es gibt aber für jede $O\text{-}Machine_n$ eine noch stärkere $O\text{-}Machine_{n+1}$, die das Halteproblem der schwächeren Maschine lösen kann. So erstreckt sich von der Turingmaschine mit leerem Orakel Ø aus eine Hierarchie von in Bezug auf Berechenbarkeit immer stärkeren *O-Machines*.[159]

Bezeichnend ist, dass das Orakel laut Turing selbst keine Maschine sein kann: „We shall not go further into the nature of this oracle apart from saying that it cannot be a machine."[160] In den Worten Hodges': „In fact, the whole point of Turing's oracle is that it describes a non-mechanical logical element."[161] Eine Prozedur, die den von der Turingmaschine definierten Bereich übersteigt, kann keine mechanische Prozedur sein. Das Orakel mag auf jede beliebige Art und Weise funktionieren – und diese Beliebigkeit ist ja allein der Grund, warum Turings *O-Machines* das Paradigma für Hypercomputer darstellen können – laut Turing ist dennoch klar, dass das Orakel nicht nach einer mechanischen Prozedur arbeitet: Es ist keine Maschine.

[158] Vgl. Turing 1939, 173.
[159] Ord 2002, 16.
[160] Turing 1939, 172f.
[161] Hodges (Internetquelle).

4.2.2 Die Accelerating Turing Machine

Damit ist klar, dass auch für Turing die Turingmaschine nicht nur eine Maschine, sondern die Grenze aller Maschinen ist. Dies hindert die Exegeten aus dem Bereich der Hyper-computation aber nicht daran, Maschinenmodelle zu entwickeln, die unberechenbare Funktionen wie das Halteproblem lösen sollen. Eine dieser Maschinen ist die *Accelerating Turing Machine*. Ihre Konzeption geht auf Copeland, aber ebenso auf Stewart[162] zurück. Die *Accelerating Turing Machine* ist, auch wenn einige im Diskurs das zumindest zeitweise anzweifelten,[163] laut Copeland eine Turingmaschine, und damit eben auch eine Maschine.
Damit diese Turingmaschine unberechenbare Super-Tasks ausführen kann, greift das Konzept auf die bereits bekannte Russell-Blake-Weyl Formel zurück (vgl. Kapitel 4.1.1). Entsprechend benötigt die *Accelerating Turing Machine* für den ersten Operationsschritt eine halbe Minute, für den zweiten Schritt nur noch halb so lange, nämlich eine viertel Minute, für den dritten Schritt eine achtel Minute, usw. Wie bereits gezeigt wurde, kann die Maschine so unendlich viele Schritte ausführen und wird damit fertig sein, bevor die Minute zu Ende ist. Wie Copeland schreibt, ist die *Accelerating Turing Machine* für die Benutzung in einem Newtonschen Universum gedacht, wie es zur Zeit von Babbages *Analytical Engine* noch allgemein angenommen wurde. In einem Einsteinschen Universum, in dem die Lichtgeschwindigkeit eine unüberbietbare Grenze bildet, könnte diese Maschine nicht wie vorgesehen unbegrenzt beschleunigen.
Die praktische Funktionsweise der *Accelerating Turing Machine* ist – zumindest in der Theorie – die folgende: Der menschliche Benutzer hat sich an die Maschine zu stellen und das eine Ende ihres Bandes in der Hand zu halten, so dass die *Accelerating Turing Maschine* sich direkt vor ihm auf dem Band befindet. Der Kniff in Copelands Ansatz ist nun, dass dieses Band zu Beginn nur so lang ist, dass die für das zu berechnende Problem nötigen Informationen auf ihm Platz haben. Die Maschine verfügt nämlich über einen eingebauten Bandgenerator: Mit diesem wird die *Accelerating Turing Machine* das Band immer dann weiter verlängern, wenn sie weitere Kästchen für die Berechnung benötigt. Wenn es also um die Berechnung einer unendlich langen Ziffernfolge geht, wird die Maschine das Band unendlich oft verlängern. Während der Operationszeit der Maschine beschleunigt auch der Bandgenerator seine Tätigkeit gleichmäßig mit der *Accelerating Turing Machine*, so dass die Arbeit der ganzen Maschine weiterhin nach der Russell-Blake-Weyl Formel abläuft.[164] Der

[162] Vgl. Stewart 1991.
[163] Vgl. Shagrir 2003. In Copeland/Shagrir 2007 scheint Shagrir auf Copelands Linie eingeschwenkt zu sein.
[164] Vgl. Copeland 2002, 289.

Benutzer hält also das noch mehr oder weniger kurze Band fest und startet die Maschine. Es wird nach der Russell-Blake-Weyl Formel eine halbe Minute dauern, bis diese den ersten Operationsschritt vollzogen und das Band entsprechend verlängert hat. Der nächste Schritt wird nur noch 15 Sekunden dauern, der dritte nur noch 7,5 Sekunden. Der dreizehnte Schritt wird weniger als eine Hundertstelsekunde benötigen, und im selben Augenblick ist die Maschine auch schon fertig – zumindest wird es für die Wahrnehmung des menschlichen Benutzers so wirken, und das, obwohl die Maschine nach diesem zehnten Schritt noch unendlich viele weitere Schritte durchlaufen hat.[165]

Wie Copeland darlegt, beschleunigt die Maschine bei diesem Vorgang unbegrenzt, so dass sie die Fluchtgeschwindigkeit erreicht und kurzerhand aus dem Newtonschen Universum verschwindet: „Take any Eucledian coordinate system originated on any point of β's [der Körper der Maschine] trajectory: the axes specify every location in the universe and β is to be found at none of them!"[166] Niemand kann also sagen, wo die Maschine nach der Lösung des Halteproblems sein wird. Ganz wie Benacerrafs Djin umgeht die *Accelerating Turing Machine* so Thomsons Paradoxon: Die Frage, in welchem Zustand die Maschine nach Ablauf des Super-Tasks sein wird, kann nicht beantwortet werden, weil die Maschine das Universum verlassen hat und ihr Zustand zwangsläufig unbekannt bleiben muss. Der Benutzer aber hält nach der ersten Minute und dem endgültigen Verschwinden der Maschine das nun endlose Band in der Hand, darauf die Lösung des berechneten Problems.

Die *Accelerating Turing Machine* wäre beispielsweise in der Lage, den Dezimalausdruck von π in voller, und das heißt unendlicher Länge auszugeben. Das aber wäre konzeptuell auch mit einer gewöhnlichen Turingmaschine möglich, da π eine berechenbare Zahl ist. Dies gilt allgemein, da die Turingmaschine selbst doch schon in der Lage ist, unendliche Dezimalausdrücke anzuschreiben, zumindest solange sie als theoretisches Konstrukt betrachtet wird. Ist die *Accelerating Turing Machine* damit nur der Versuch, die „Prinzipschaltung"[167] Turingmaschine ins Reale zu übersetzen? Zielt dieses Konzept also gar nicht auf Hypercomputation ab, sondern ausschließlich auf die Differenz zwischen symbolisierten und realen Maschinen? Es stellt sich die grundsätzliche Frage, welche Auswirkungen es in Bezug auf Berechenbarkeit mit sich bringt, wenn unendlich viele Operationsschritte in endlicher Zeit vollzogen werden können.

---

[165] Copeland 2002 setzt für den ersten Schritt der *Accelerating Turing Machine* 1 Sekunde an. Für die Funktionsweise der Maschine nach der Russell-Blake-Weyl Formel und die Argumentation spielt dieser Unterschied keine Rolle.
[166] Ebd., 289.
[167] Kittler 1990, 244.

Die Antwort ist, dass die *Accelerating Turing Machine* tatsächlich mehr leistet als die gewöhnliche Turingmaschine. Wie die *O-Machines* ist sie in der Lage, das Halteproblem zu lösen. Der Grund dafür ist, dass das Argument Turings, diese *reductio ad absurdum*, für unendlich beschleunigende Turingmaschinen nicht mehr gilt:

> [I]n the original proof, a new machine was created that solves the halting function for a given machine and its input, halting if and only if that machine does not. Such an accelerated Turing machine cannot be constructed. If an accelerated Turing machine is set to compute the halting function and its input specifies a non-halting computation, this machine cannot determine this and then halt. Determining that this machine does not halt involves the accelerated Turing machine looping. Thus the paradoxical machine cannot be created and the argument for the contradictory nature of accelerated Turing machines fails.[168]

Die Präzisierung des *Zeitverhaltens* der Turingmaschine schlägt also bis auf die Ebene dessen durch, was sich *logisch* über die Mächtigkeit dieser Maschine in Bezug auf Berechenbarkeit schließen lässt. Das bedeutet auch: *Für die Operationsschritte der Turingmaschine muss implizit eine gewisse minimale Dauer angenommen werden*. Wären die Operationsschritte nämlich beliebig schnell ausführbar, gäbe es keinen Grund, warum die Turingmaschine nicht auch nach der Russell-Blake-Weyl Formel ablaufen sollte. Dann aber läge keine Turingmaschine mehr vor, wie Turing sie 1936 definiert hat, sondern eine mächtigere Maschine, die ganz wie die *Accelerating Turing Machine* in der Lage wäre, das Halteproblem zu lösen. Ein Konstrukt also, das in der Ansicht Turings gar keine Maschine mehr wäre.
Wenn für die Operationen der Turingmaschine aber eine gewisse minimale Dauer angenommen werden muss, dann erhält Turings einleitender Satz von 1936 eine erweiterte Bedeutung. Er schreibt dort: „The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means.“[169] Sicherlich bezieht sich Turing mit der Wendung „by finite means“ bewusst nur auf den Umstand, dass die Tabellen, also die Algorithmen der Turingmaschine endlich sein müssen – während die Rechnung wie im Fall von π durchaus unendlich laufen darf. Wie der Vergleich mit der *Accelerating Turing Machine* aber zeigt, darf dieser Satz auch in Bezug auf eine notwendigerweise nur endliche Rechengeschwindigkeit der Turingmaschine verstanden werden. Die Turingmaschine als eine Maschine der Grenze und als Grenze der Maschinen scheint sich Fragen bezüglich ihres Zeitverhaltens damit weniger zu entziehen als gemeinhin angenommen.

---

[168] Ord 2002, 30. Die Verflechtung von *Satisfactoriness Problem* und Halteproblem darzustellen – an und für sich zwei verschiedene Probleme, die Ord dennoch zusammenzieht – wäre Aufgabe einer anschließenden Arbeit.
[169] Turing 1936, 58.

4.2.3 Shrinking Machines

Auch wenn Copelands *Accelerating Turing Machine* in weiten Teilen mit den physikalischen Gesetzen des 19. Jahrhunderts konform geht, wäre die als Folge der Beschleunigung auftretende Hitzeentwicklung ein kaum zu lösendes Problem. Davies' *Shrinking Machine* kommt dagegen ohne unbegrenzt schnelle Bewegungen aus. Das bedeutet, sie benötigt lediglich endliche Energieressourcen und kann auf die Verwendung unendlich starker Materialen verzichten. Um mögliche Einwände bezüglich der Eigenschaften von Elektrizität und Magnetismus auszuschließen, nimmt Davies an, dass seine Maschine rein mechanisch funktioniert. Damit stellt er sicher, dass der Apparat, der ihn selbst an die *Analytical Engine* von Charles Babbage erinnert, tatsächlich vollständig mit der um 1850 bekannten Physik konsistent ist.[170] Obwohl die Möglichkeit solcher Modelle für ein Newtonsches Universum schon länger bekannt ist, ist Davies' Entwurf nach eigenem Bekunden der erste dieser Art.[171]

Das Konzept der *Shrinking Machine* macht sich eine andere Eigenschaft des Newtonschen Universum zunutze als die *Accelerating Turing Machine* mit ihrer unendlichen Beschleunigung. Davies referenziert mit Newtonschem Universum auf „a universe obeying Newton's laws in which matter may be subdivided more and more finely while retaining the same properties."[172] Nach der Grundidee von Thomsons unendlich teilbarer Schokolade[173] ist die *Shrinking Machine* in der Lage, immer kleinere Versionen von sich selbst zu erzeugen.

Dementsprechend entwirft Davies eine Maschine M_1. Diese enthält einen endlichen Computer von ähnlichem Typ wie die Maschine Babbages, d. h. von rein mechanischem Aufbau. Dieser Computer verfügt wiederum über eine in einem spezifischen Takt c_1 arbeitende *clock* und einem Speicher mit der Größe m_1 Bytes. Außerdem, und das ist der Kniff, besitzt die Maschine M_1 eine eigene Roboterfabrik, in der sie eine neue Version von sich selbst samt neuer Fabrik erstellen kann, nämlich die Maschine M_2. Diese neue Maschine M_2 unterscheidet sich von der sie produzierenden Maschine M_1 dahingehend, dass sie über einen informationstheoretisch gesehen doppelt so großen Speicher verfügt ($m_2 = 2m_1$). Zudem sind ihre Komponenten 16 mal kleiner als die der Maschine M_1. Daraus folgt, dass die neue

[170] Vgl. Davies 2001, 2.
[171] Vgl. ebd., 1.
[172] Ebd., 1.
[173] Oder frei nach dem Aphorismus Lichtenbergs, der da sagt: „Ich glaube nicht, daß es ganz unmöglich wäre, daß ein Mensch ewig leben könne; denn *immer abnehmen* schließt den Begriff von *aufhören* nicht notwendig in sich." Lichtenberg 1990, 375.

Maschine M_2 nur 1/8 der Größe der alten Maschine M_1 hat, also um ein Vielfaches kleiner ist als ihr Vorgänger. Das bedeutet, dass die Signalwege in Maschine M_2 kürzer sind als in der Maschine M_1, so dass die Signale trotz gleichbleibender Übertragungsgeschwindigkeit in der neuen Maschine nur noch höchstens 1/8 der Laufzeit brauchen:[174] „Qualitatively speaking M_2 is smaller, faster and more powerful than M_1.“[175] Ebenso ist nun auch M_2 in der Lage, eine ebenfalls um 7/8 kleinere Version ihrer selbst zu erstellen, die Maschine M_3, usw. Jede Maschine M_n produziert also bei Bedarf die Maschine M_{n+1} als kleinere Version ihrer selbst. Aufgrund der stetigen Verkleinerung aller aufeinander folgenden Maschinen existiert für die räumliche Ausdehnung auch unendlich vieler Maschinen ein endlicher Grenzwert. Tatsächlich würde eine unbegrenzte Reihe solcher Maschinen nur etwas mehr Raum einnehmen als die ursprüngliche Maschine M_1. Zudem produziert jede der Maschinen M_n ihren Nachfolger M_{n+1} aufgrund der kürzeren Signal- und Transportwege auch schneller, nämlich in höchstens 1/8 der vorher benötigten Zeit. Das bedeutet, dass unendlich viele solcher Maschinen in endlicher Zeit produziert werden können.[176]

Interessanterweise begründet Davies seine Idee der Konstruktion von Maschinen, die wiederholt eine kleinere Version von sich selbst erstellen, historisch: „Each generation of tools has been used to produce the next one, and as time has passed the tools have got steadily smaller, faster and more accurate.“[177] Ganz im Sinne Kittlers bezieht er sich damit auch auf die tatsächliche Entwicklung der Computer: „Similar remarks apply to computer design and performance over the period; this is a relevant observation since each generation of computers and machine tools is used to design and build the next.“[178]

Um Super-Tasks auszuführen und unberechenbare Probleme zu lösen, müssen die Maschinen zusammenarbeiten. Dafür verfügen sie über einen Kommunikationskanal: Jede Maschine kann über die dazwischen liegenden Maschinen mit jeder beliebigen anderen Maschine kommunizieren. Dabei wird über Listen festgelegt, welche Menge an Daten eine Maschine einer anderen übertragen kann. Dies ist notwendig, da die Speicher der Maschine bei jedem Schritt doppelt so groß werden, M_n also nur die Hälfte der Daten von M_{n+1} speichern kann.[179]

[174] Vgl. Davies 2001, 2.
[175] Ebd., 2.
[176] Vgl. ebd., 3.
[177] Ebd.
[178] Ebd. Kittler schreibt von dieser Entwicklung unter anderem in 1996a, 131 und 1994a, 186-189.
[179] Vgl. Davies 2001, 3f.

Davies gibt unter anderem das Halteproblem als eine unberechenbare Funktion an, die mittels seiner Maschine(n) über diesen Kommunikationskanal gelöst werden kann. Die Operationsweise ist dabei die folgende: Die Beschreibungsnummern n der zu testenden Turingmaschinen werden in aufsteigender Reihenfolge getestet, beginnend mit der Beschreibungsnummer $n = 1$ auf der Maschine M_n, d. h. auf der ersten Maschine M_1. Diese Maschine führt die Turingmaschine mit der Beschreibungsnummer $n = 1$ aus und prüft, ob diese anhält. Stoppt sie, so ist diese Beschreibungsnummer erledigt und die nächste Maschine M_{n+1} beginnt mit der Ausführung von n+1. Füllt sich aber der Speicher der gerade rechnenden Maschine M_n oder hat diese Maschine 2n Schritte ausgeführt bevor das Programm stoppt, so wird dieses an die nächst kleinere Maschine M_{n+1} weitergegeben. Liegt mit n also die Beschreibungsnummer einer Turingmaschine vor, die nicht anhält, dann wird diese die unendliche Reihe der Maschinen herabgereicht. Da die Rechenoperationen und die Konstruktion der rechnenden Maschinen nach der obigen Beschreibung nur eine endliche Zeitspanne in Anspruch nehmen, erhält die Maschine M_1 nach Ablauf einer endlichen Zeitspanne Nachricht über den Kommunikationskanal, dass es sich bei n um die Beschreibungsnummer einer Turingmaschine handelt, die nicht anhält.[180] Da die *Shrinking Machine* ganz wie die *Accelerating Turing Machine* die nachfolgenden Operationsschritte schneller ausführt als die vorhergehenden – auch wenn sie dies auf der Basis eines gänzlich anderen Konzepts vollzieht und nicht nach jedem einzelnen Operationsschritt beschleunigt, sondern beim Wechsel von einer Maschine auf die andere – greift auch hier Turings *reductio ad absurdum* nicht, so dass der logische Ausschluss des Halteproblems aus dem Bereich des Berechenbaren sich seiner Prämissen versagt sieht.[181] Dass in einem Einsteinschen Universum das Plancksche Wirkungsquantum der unendlichen Verkleinerung eine unüberschreitbare Schranke setzt, ändert nichts dran, dass die *Shrinking Machine* nach den Gesetzen des Newtonschen Universum, wie es die in der Geschichte der Wissenschaft für eine nicht unerhebliche Zeitspanne als allein gültig erachtet wurde, funktionstüchtig gewesen wäre.[182] Die *Shrinking Machine* zeigt daher in hervorragender Weise, dass Hypercomputer unter bestimmten Bedingungen tatsächlich möglich sind. Damit steht sie an der Nahtstelle zwischen rein logischen Überlegungen über Möglichkeit und Unmöglichkeit von Super-Tasks, wie denen Thomsons und Benacerrafs, und aktuellen Versuchen, Hypercomputer tatsächlich zu realisieren.

[180] Vgl. ebd., 5f.
[181] Eine detaillierte Analyse, unter welchen Bedingungen und für welche Maschinen Turings Beweisführung Gültigkeit verlangen darf, wäre ein weiteres mögliches Thema für eine kritische Fortsetzung dieser Arbeit.
[182] Vgl. ebd., 1 und 6.

4.2.4 Non-standard Quantenrechner

Wie Copeland und Proudfoot behaupten, ist das Berechnen von im klassischen Sinne unberechenbaren Funktionen nur eine Frage der Zeit und der geeigneten Technologie.[183] Wie Ord anführt, ist es gut möglich, dass diese Hypercomputer, sollten sie tatsächlich umsetzbar sein, exponentiell langsamer arbeiten als die digitalen Maschinen der heutigen Zeit.[184] Es ist ebenso möglich, dass sie schneller arbeiten, wie es ja für die *Accelerating Turing Machine* und die *Shrinking Machine* den Anschein haben musste. Damit soll noch einmal betont sein, dass die Arbeitsgeschwindigkeit, die eine reale Maschine am Ende empirisch messbar aufweist, zumindest vordergründig keine Rolle zu spielen scheint, wenn es um die Frage geht, ob es sich dabei um einen Hypercomputer handelt, oder nicht. Das wendet sich allerdings um, sobald die Maschine Rechenoperationen mit unendlich kleiner Dauer ausführen kann, wenn die Zeit der Maschine also das finite Maß endgültig unterschreitet und sie in gewissem Sinne selbst medientechnisch *unmessbar* wird. Mit einem unendlich kleinen Zeitmaß entzieht sich eine Maschine, wie am Beispiel der *Accelerating Turing Machine* gezeigt wurde, selbst dem logischem Beweis für die Unberechenbarkeit des Halteproblems. Dass eine solche Maschine als Anachronismus für ein Newtonsches Universum möglich wäre, wurde anhand der *Shrinking Machine* dargestellt. Es existieren aber auch aktuell Versuche, Hypercomputer real umzusetzen. Dafür bietet sich in den Augen einiger die Quantenmechanik in besonderer Weise an. Die Idee dahinter ist die folgende:

> Every existing general model of computation is effectively classical. That is, a full specification of its state at any instant is equivalent to the specification of a set of numbers, all of which are in principle measurable. Yet according to quantum theory there exist no physical systems with this property. The fact that classical physics and the classical universal Turing machine do not obey the Church-Turing principle in the strong physical form is one motivation for seeking a truly quantum model. The more urgent motivation is, of course, that classical physics is false.[185]

Diese Aussage hat allerdings nur Sinn, wenn Quantenrechner nicht länger im Modell des *standard quantum computing* konzipiert werden. Dieses entspricht, ganz wie die Turingmaschine, dem Weltbild der klassischen Physik, das sich – wie Deutsch es so drastisch wie eindeutig formuliert – eben als falsch herausgestellt hat.[186] Entsprechend gibt es Ansätze, wie die Quantenmechanik genutzt werden könnte, um zuvor unberechenbare Probleme mit technischem Gerät medial vermittelt doch zu lösen.

---

[183] Vgl. Copeland/Proudfoot 1999.
[184] Vgl. Ord 2002, 42.
[185] Deutsch 1985, 102.
[186] Vgl. ebd.

Im ersten Jahrzehnt dieses Jahrtausends war es vor allem der Vorschlag von Kieu,[187] der eine technologische Umsetzung der Hypercomputation in greifbare Nähe zu rücken schien: Kieu legte seinem Ansatz eben nicht das Rechnen mit *q-bits* und Operationen an elementaren logischen Gattern zugrunde,[188] sondern griff stattdessen auf die adiabatische (d. h. eine wärmedichte) Kühlmethode von Farhi zurück, die dieser im Jahr 2000 als ein alternatives Modell für Quantenberechnung entwickelt hatte.[189] Die vorliegende Untersuchung wird an dieser und vergleichbaren Stellen nicht in die technischen Details gehen. Die Arbeiten Kieus zeigen allerdings dessen Überzeugung, ein technisch realisierbares Verfahren gefunden zu haben, mit dem sich das Zehnte Problem Hilberts, das im Wesentlichen dem oben beschriebenen Halteproblem äquivalent ist (vgl. Kapitel 2.2.2), lösen lassen sollte.[190]

Smith, Hodges sowie Hagar und Korolev[191] haben dieser Hoffnung auf mathematischer und physikalischer Ebene den Boden entzogen, indem sie kleinere und größere Fehler in Kieus physikalischer und mathematischer Argumentation zeigen konnten. Eine Umsetzung von Kieus Projekt des adiabatischen Quantencomputers ist aufgrund dieser Kritik sehr unwahrscheinlich.[192] Dessen ungeachtet bietet gerade Hodges' Kritik die Gelegenheit, sich über das grundlegende Konzept in Kieus Ansatz klarer zu werden.

So schreibt Hodges: „Kieu's claim is that for the infinite search required in the Hilbert problem, the speed-up can actually turn an infinite time into a finite time."[193] Kieu macht es in seinen Texten zwar nicht explizit, aber laut Hodges ist klar, dass „[…] the basis of Kieu's claim is that an infinite amount of information can be compressed into a finite physical system."[194] Da bereits gegeben ist, dass der verwendete Speicher von lediglich endlich großen Gesamtmaßen sein muss, kann es, wenn tatsächlich unendlich viel Information im speichernden System Platz haben soll, kein unteres Limit der Messgenauigkeit geben. In anderen Worten: Um eine unendliche Menge Information in einem endlichen physikalischen System unterzubringen muss *unendliche Präzision* geschaffen werden.

---

[187] Vgl. für diesen Vorschlag Kieu 2003a; 2003b; 2003c; 2003d und 2004. Vgl. zu den damit verbundenen Hoffnungen auch Hodges 2006, 1.

[188] Andere alternative Ansätze nutzen das Modell der Malament-Hogarth Raumzeit, um Konzepte für Hypercomputer zu entwickeln (vgl. Etesi/Németi 2002, Hogarth 1992 und 1994, sowie Wischik 1997). Calude und Pavlov (2002) untersuchen die Möglichkeit für Hypercomputation mittels der Quantenmechanik in einem breiteren Kontext und beziehen dabei explizit Stellung gegen Feynmans frühere Ansicht, dass diese nicht möglich sei (vgl. Feynman 1982).

[189] Vgl. Farhi/Goldstone 2000.

[190] Zur Äquivalenz von Hilberts Zehntem Problem und dem Halteproblem vgl. besonders Davis 1958.

[191] Vgl. Smith 2006; Hodges 2006 und Hagar/Korolev 2007.

[192] Vgl. Hagar/Cuffaro 2015 (Internetquelle).

[193] Hodges 2006, 2.

[194] Ebd., 5.

Eine theoretische Möglichkeit, mittels unendlicher Präzision einen solchen unendlichen Speicher mit lediglich endlichen Gesamtmaßen zu realisieren, liefern die schon bekannte Russell-Blake-Weyl Formel und deren Derivate, wenn sie einmal mehr nicht temporal, sondern spatial ausgelegt werden:

Speicher mit finiter Ausdehnung und unendlich vielen Speicherstellen.[195]

Jede Speicherstelle nimmt nach dieser Formel nur einen Bruchteil der vorhergehenden ein. So haben in einem endlichen Raumstück unendlich viele binäre Speicherstellen Platz, die in der obigen Abbildung nur der besseren Erkennbarkeit halber alternierend in schwarz und weiß ausgefüllt sind. So lässt sich eine beliebige unendliche Binärzahl speichern, wie zum Beispiel die Lösung des Halteproblems τ. Die unendliche Präzision wird hierbei erst gegen Ende des Speichers notwendig, wenn die letzte Speicherzelle unendlich klein wird.
Eine zweite Möglichkeit, einen unendlichen Speicher von endlicher Ausdehnung zu konstruieren, ist, die unendliche Präzision nicht durch unendlich viele Speicherstellen zu erzeugen, sondern sie in einem einzigen exakten Schnitt zu konzentrieren:

Speicher mit finiter Ausdehnung und einem unendlich präzisen Schnitt.

Während der ersten Methode nur rationale Zahlen zugrunde liegen – dies ganz einfach deswegen, weil sich das Verhältnis jeder Speicherstelle zum Gesamtspeicher nach der Russell-Blake-Weyl Formel als ganzzahliges Verhältnis darstellen lässt[196] – liest die zweite Methode den einzelnen Schnitt im Speicher als reelle Zahl aus.[197] Da reelle Zahlen einen unendlichen Dezimalausdruck haben, muss dieser Schnitt mit unendlicher Präzision im unendlich exakt bemessenen Speicher gesetzt werden, um eine beliebige unendlich lange Zahl zu speichern. Während im ersten Beispiel nicht die exakte Position der Schnitte, sondern der Inhalt der durch sie definierten Speicherstellen die Grundlage der Methode bildet, kommt im zweiten Beispiel der Position des unendlich feinen Schnittes die entscheidende Rolle zu.

[195] Beide Abbildungen auf dieser Seite aus Ord 2002, 25. Diese Unterscheidung bietet zudem einen exzellenten Ansatzpunkt, um über die Begriffe analog und digital, kontinuierlich und diskret sowie symbolisch und real in Bezug auf Speicher und darüber hinaus nachzudenken, evtl. unter Bezugnahme auf Ernst/Maibaum/Völz 2016.
[196] Nämlich der Speicher selbst als 1, die erste Speicherzelle als ½ des Gesamtspeichers, die zweite als ¼, usw.
[197] Vgl. Ord 2002, 25.

Kieus Ansatz kann sich selbstredend nicht auf solche rein theoretischen Modelle berufen. Dessen Stichhaltigkeit hängt notwendig davon ab, ob reale physikalische Systeme mit unendlicher Präzision aufgestellt und beständig gemacht werden können.[198] Hodges greift auf die Fourieranalyse zurück, um das Phänomen der unendlichen Präzision deutlicher zu illustrieren:

> The basic difficulty here is not essentially quantum-mechanical. It arises in just the same way if we consider encoding an infinite amount of data into a real-valued continuous curve on [0,1] through Fourier decomposition. One may imagine a wave on a string with unboundedly high harmonics, with the amplitude of the *N* th harmonic encoding the *N* th piece of data. Unboundedly high harmonics correspond to unboundedly short wavelengths, so that detecting *every* harmonic requires infinite precision. Measurement apparatus incapable of a resolution better than wavelength 1/*N*, will be deaf to harmonics higher than the *N* th, and lose all the data after the *N* th piece.[199]

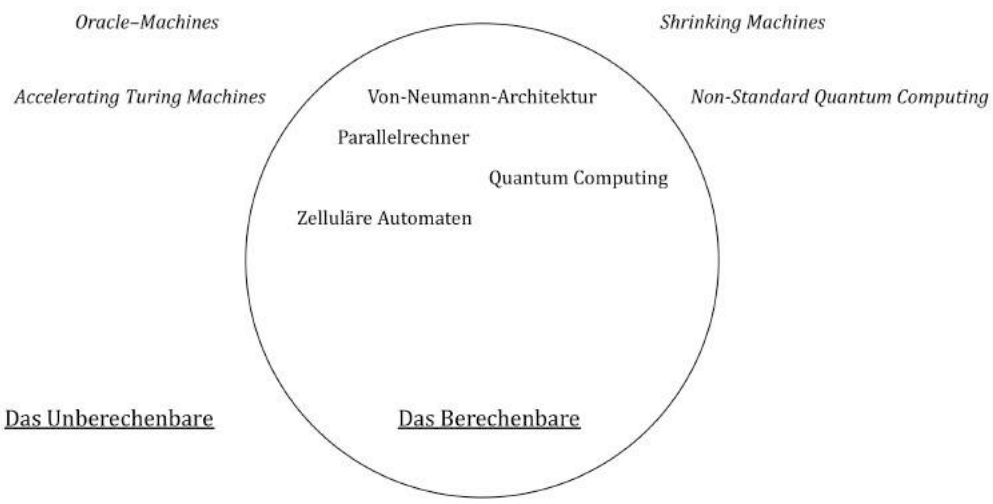
Auch wenn die reale Umsetzbarkeit von Kieus Ansatz bereits wieder vom Tisch zu sein scheint, lässt sich Folgendes konstatieren: Ähnlich wie bei Davies' *Shrinking Machine* wird hier der Raum unendlich fein unterteilt – auch wenn dem bei Kieu kein klassischer Raumbegriff mehr zugrunde liegt. Dieses Anliegen, die Unendlichkeit im Realen zu zähmen, findet sich auch in vergleichbaren aktuellen Ansätzen. So vertritt Hogarth die Ansicht, die generelle Relativitätstheorie erlaube es ähnlich der *Accelerating Turing Machine*, eine unendliche Anzahl von Operationsschritten in endlicher Zeit auszuführen.[200] Caludes und Pavlovs Modell sieht dagegen lediglich endlich schnelle Operationsschritte vor, fußt aber darauf, dass eine unendliche Anzahl dieser Schritte parallel stattfindet.[201] Ist also das Unendliche für Gauss noch lediglich eine *façon de parler*,[202] geht es in den aktuellen Konzepten für Hypercomputation darum, dieses Unendliche mit endlichen Mitteln im Realen zu erzwingen.[203] Sollten diese Versuche tatsächlich von Erfolg gekrönt sein und im Sinne Copelands und Proudfoots die richtige Technologie zu ihrer Umsetzung gefunden werden, so stünden ab da technische Medien bereit, um auch unberechenbare Funktionen zu lösen. Dies würde allerdings bedeuten, dass hochtechnische Medien endgültig zu Black Boxes würden. Noch ist es menschenmöglich, die Operationen eines digitalen Rechners mit Papier und Bleistift nachzuvollziehen, auch wenn diese Aufgabe vielleicht auf eine ganze Zivilisation verteilt werden müsste. Ein Hypercomputer allerdings, der unendliche Aufgaben in endlicher Zeit erledigt, würde diese Tür des menschlichen Nachvollzugs endgültig zuwerfen.

---

[198] Vgl. Hodges 2006, 5.
[199] Ebd., 6.
[200] Vgl. Hogarth 1992 und 1994, sowie Ord 2002, 25.
[201] Vgl. Calude/Pavlov 2002.
[202] Gauss 1831, 269 (Internetquelle).
[203] Eine Betrachtung von *Random Machines* muss in dieser Arbeit unterbleiben. Es wäre aber von großem Interesse zu prüfen, inwiefern die hier aufgestellten Behauptungen auch für diese Geltung verlangen können.

4.2.5 Unendliche Medien

Wie gezeigt wurde, liegen die vorgestellten Konzepte für Hypercomputation tatsächlich außerhalb des Bereichs des Berechenbaren. Zwar sind Quantenrechner mittels des *standard quantum computing* nur zur Abarbeitung von berechenbaren Funktionen in der Lage. Wie die soeben umrissenen Modelle aus dem Bereich des *non-standard quantum computation* allerdings zeigen, gibt es für die Theorie des Quantenrechnens sowohl Konzepte für den Bereich des Berechenbaren, als auch darüber hinaus.



Die Konzepte aus dem Bereich der Hypercomputation reichen über den Bereich des Berechenbaren hinaus.

Solche Hypercomputer ließen sich medientheoretisch als Versuche einer „[…] totalen Mobilisation von Zeitmächtigkeiten in Kommunikationsmedien“[204] verstehen, um eine Formulierung von Ernst zu entlehnen. Dies gilt nicht nur für die konzeptuell auf einer unendlich schnellen Rechengeschwindigkeit beruhenden Modelle. Da es bei Hypercomputation stets darum geht, unendliche Super-Taks in einem endlichen Zeitintervall abzuschließen, d. h. Aufgaben, deren Ausführung bei endlicher Geschwindigkeit unendlich viel Zeit benötigen würde, in ein endliches Zeitintervall zu komprimieren, kann diese Formulierung für sämtliche Modelle der Hypercomputation Geltung verlangen.
Im Folgenden wird gezeigt, inwiefern die explizierten Konzepte aus dem Bereich der Hypercomputation zu einem klareren Begriff sowohl von Turingmaschine als auch von Berechenbarkeit beitragen. Dabei wird auch deutlich, dass die Turingmaschine nur vergleichsweise unklar zu dieser „totalen Mobilisation von Zeitmächtigkeiten“ zu zählen ist.

[204] Ernst 2012a, 33.

4.3 Implikationen für die Turingmaschine

4.3.1 Die Turingmaschine als theoretisches Konstrukt

Dass Turing die Turingmaschine auf dem Papier entwirft, bedeutet noch nicht, dass es sich bei dieser zwangsläufig um ein *rein* theoretisches Konstrukt handeln muss. Zahlreiche Maschinen werden am Reißbrett entworfen, gerade um dann real gebaut zu werden.[205] Was die Turingmaschine aber dennoch als rein theoretisches Konstrukt ausweist, ist der Umstand, dass sie Zahlen mit unendlichem Dezimalausdruck wie π oder *e* berechnet. Gerade vor dem Hintergrund der Hypercomputation lohnt es sich, diesen Zusammenhang näher zu betrachten.
Wie kann die Turingmaschine unendliche Zahlen ausgeben? Kittler antwortet auf diese Frage mit der Behauptung, die Turingmaschine sei eine „[…] Maschine mit […] grenzenloser Rechengeschwindigkeit […].“[206] Wie sich aber gezeigt hat, ist eine Turingmaschine, die unendlich schnell rechnet, keine Turingmaschine mehr, die Turings Definition von Berechenbarkeit erfüllen würde. Kittlers Behauptung ist also falsch. Seine gegenteilige Annahme, „Turings Prinzipschaltung von 1936 […] [sei] unendlich langsam […]“[207] ist dagegen zwar nicht falsch, aber in diesem Kontext schlicht sinnlos.[208]
Richtig ist, dass die Turingmaschine, damit sie Zahlen wie π oder *e* in voller Ausführlichkeit berechnen kann, zwei Ressourcen benötigt: Unendlich viel Zeit und unendlich viel Speicher. So schreibt Turing in Bezug auf letzteren: „Es war für diese theoretische Argumentation wesentlich, daß der Speicher unendlich sein sollte. Es kann leicht gezeigt werden, daß die Maschine andernfalls nur periodische Oszillationen ausführen kann.“[209] Diese Forderung ist erfüllt durch das unendliche Band, mit dem Turings Maschine versorgt wird. Dass die Turingmaschine aber auch über unendlich viel Zeit verfügen muss, dieser Forderung trägt kein besonderes Bauteil in Turings Design Rechnung.[210] Der Grund, aus dem heraus der Turingmaschine unendlich viel Zeit für ihre Rechnungen zur Verfügung steht, ist vielmehr der, dass sie nicht in einer entropischen, sondern in einer logischen Zeit situiert ist. Die einzelnen Schritte der Turingmaschine müssen zwar zwangsläufig eine minimale Dauer haben. Da es aber keinen logischen Widerspruch darstellt, *unendlich lange* zu rechnen, ist auch die Ausgabe sämtlicher Stellen von π für die Turingmaschine logisch möglich.

---

[205] Auch wenn der Signifikant ‚Reißbrett' heute nur mehr als Metapher für digitale CAD-Modelle herhält.
[206] Kittler 1993, 294f.
[207] Kittler 1992, 264.
[208] Kittler gibt auch technisch exakte Beschreibungen der Turingmaschine, wie gezeigt aber nicht immer.
[209] Turing 1947, 186.
[210] Denn welches besondere Bauteil sollte die Forderung nach unendlich viel Zeit auch erfüllen?

Hier liegt die so triviale wie eindeutige Differenz zwischen der Turingmaschine und allen ihren realen Umsetzungen, seien diese als Harvard- oder von Neumann-Architektur, als Parallelrechner, als universale zelluläre Automaten oder nach dem *standard quantum computing* umgesetzt. Keine der realen Maschinen, die aller Miniaturisierung zum Trotz die Lebenswelt immer aufdringlicher prägen, kann praktisch vollziehen, was die Turingmaschine theoretisch leisten kann. Die dafür nötigen Ressourcen wie Speicher, Energie oder Haltbarkeit werden angesichts der unendlichen Länge des Dezimalausdrucks natürlicherweise erschöpft sein, bevor π oder eine andere im Sinne Turings berechenbare Zahl ausgeschrieben ist.
Nun existieren aber, wie gezeigt wurde, durchaus Konzepte aus dem Bereich des *non-standard quantum computing*, mittels derer unendliche Zahlen wie π ausgebbar sein sollen und deren reale Umsetzung zumindest von einigen erhofft und erwartet wird. Sollte diese Umsetzung gelingen, hätte sich damit nicht auch die Turingmaschine als real umsetzbar erwiesen? Wäre ihr dann nicht der Status einer rein logischen Maschine abzusprechen? Die Antwort ist nein, denn, wie sich eben gezeigt hat, kann es keine Maschine geben, die einen solchen Super-Task abschließen und dabei Turingmaschine bleiben kann. *Eine unendlich schnelle Rechenzeit lässt sich zwar vielleicht technisch umsetzen. Die Umsetzung einer unendlich langen Rechenzeit aber ist nach gegenwärtigem Stand der Physik nicht real, sondern nur logisch möglich.* In diesem Sinne existiert die Turingmaschine lediglich als theoretische Maschine innerhalb einer „theoretischen Argumentation", situiert im Bereich des Symbolischen – im Sinne Kittlers – und daher nur einer symbolischen Zeit unterworfen.

4.3.2 Die Operativität der Turingmaschine

Soll die Turingmaschine also in den Bereich des Realen übertragen werden, so muss sie entweder hinter ihren Möglichkeiten zurückbleiben oder diese sprengen – sie kann in Bezug auf Berechenbarkeit nicht sie selbst bleiben. Das bedeutet: *Effektiv* kann die Turingmaschine einen Super-Task, wie ihn die Berechnung von π darstellt, ausführen.[211] Soll die Turingmaschine aber *effizient* sein, d. h., den Super-Task tatsächlich innerhalb einer endlichen Zeitspanne abschließen, so muss ihre Umsetzung entweder an den Zwängen des Realen

[211] Vgl. zu Super-Tasks (im Unterschied zu Hypercomputation, ein Unterschied, dem hier nur flüchtig Rechnung getragen wurde) auch den bereits zitierten und hervorragenden Artikel der *Stanford Encyclopedia of Philosophy* (s. Internetquellen Manchak/Roberts 2016). Weitere Modelle für Super-Tasks mit interessanten Implikationen sind auch das dort genannte Littlewood-Ross Paradoxon, das u. a. mittels eines Einmachglases vonstatten geht (vgl. John Littlewood (1953) als ein Unendlichkeitsparadoxon, erweitert von Sheldon Ross (1988)), sowie der *Beautiful Supertask* von Pérez Laraudogoita (1996; 1998), einem klassisch mechanischen Super-Task anhand von elastischen Stößen, die durch unendlich viele Partikel weitergegeben werden. Vgl. auch die Aufzählung verschiedener Super-Tasks in Earman/Norton 1996.

scheitern oder sie verlässt im selben Zug unweigerlich den Rahmen der Berechenbarkeit und ist keine Turingmaschine mehr.[212] Will man das genau nehmen, muss man hinzufügen: Die universale Turingmaschine ist ein theoretisches Konstrukt, nicht nur aus den angegebenen Gründen, sondern auch, weil sie nicht als sie selbst, sondern nur als eine spezielle Turingmaschine, deren Funktionsweise sie übernimmt und ausführt, *operativ* werden kann.[213] Gewisse spezielle Turingmaschinen, namentlich all die, die nach einer endlichen Anzahl von Operationsschritten anhalten, lassen sich natürlich als reale Maschinen umgesetzt denken. Da Turing aber nur solche Turingmaschinen als befriedigend betrachtet, die unendliche Zahlen ausgeben, gilt für alle diese Turingmaschinen, das sie sich von realen Maschinen um den Faktor Unendlich unterscheiden, was die Länge der von ihnen ausgebbaren Ergebnisse betrifft.

Wie weit aber können bisherige reale Maschinen rechnen, wenn sie arithmetische und andere Probleme nicht wie unendliche Maschinen durch ein totales *brute force* lösen können?[214] Wo muss die Grenze des mit realen Maschinen Berechenbaren gezogen werden? Auf ein bereits bekanntes Beispiel bezogen wäre dies die Frage: Bis zur wievielten Stelle kann eine reale Maschine π berechnen?

Selbst wenn klar ist, dass jede reale Maschine eine unendliche Berechnung irgendwann abbrechen muss, kann sie bei ihrer Berechnung doch beliebig weit gekommen sein, solange sie dabei nur eine endliche Anzahl von Schritten vollzogen hat. Sie könnte *stets noch einen Operationsschritt mehr machen*[215] – was das Wesen der Rekursion, die der mechanischen Prozedur auf mathematischer Seite äquivalent ist, auf simple Weise fasst. Daher lässt sich keine Grenze angeben, die nicht irgendwann, unter besonderen Bedingungen und von besonders schnellen Maschinen, überboten werden könnte. So wurden und werden seit dem ägyptischen Papyrus Rhind und Archimedes mehr und mehr Nachkommastellen von π berechnet. Da endliche Formeln, und d. h. im Kontext der Turingmaschine nichts anderes als Instruktionstabellen, existieren, mit denen π vollständig berechnet werden könnte, ist es eben möglich, immer noch eine Nachkommastelle mehr auszugeben. Dass alle realen Maschinen dennoch die Berechnung stets nach einer endlichen Anzahl von ausgegebenen Stellen abbrechen müssen, liegt in der Faktizität der Welt begründet, so dass beispielsweise der Programmierer das diesbezügliche Programm eben nach einer gewissen Zeit abbricht oder der

[212] Vielleicht enthält die Behauptung, dass eine rein logische Maschine etwas berechnen kann, überhaupt schon einen medientheoretischen Kategorienfehler.

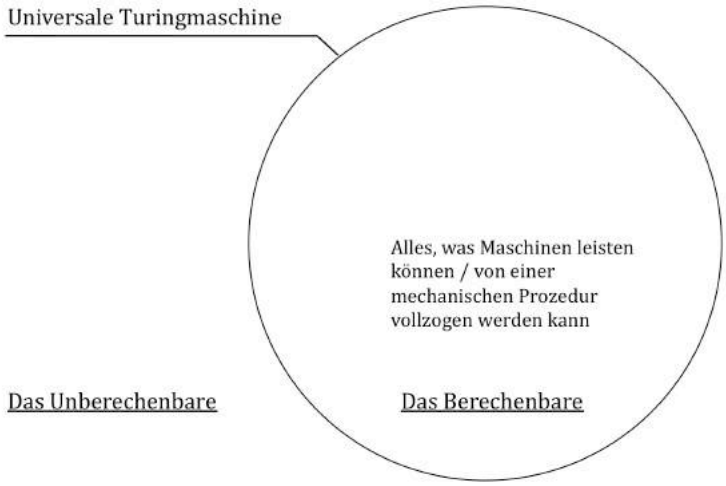
[213] Die Frage, was passiert, wenn eine universale Turingmaschine ihr eigenes Programm, das heißt, ihre eigene Beschreibungsnummer auf dem Band liest und ausführt, muss an anderer Stelle beantwortet werden.

[214] Vgl. Davies 2001, 1.

[215] Diese Formulierung ist eine Abwandlung der Formulierung van Treecks, „reale Maschinen können stets noch eine Zahl mehr berechnen."

Speicher der Maschine voll läuft und keine Finanzen mehr zur Verfügung stehen, um ihn zu erweitern. Die Berechnung unendlicher Zahlen durch reale Maschinen kann eben immer nur eine Annäherung sein. In der Praxis wird diese Annäherung so weit durchgeführt, dass die Ergebnisse für ebendiese Praxis hinreichend genau sind, so wie π beispielsweise gerundet auf 3,14 für die meisten herkömmlichen Anwendungen völlig ausreicht.

Die Grenze dessen, was mit realen endlichen Maschinen allgemein möglich ist, verläuft daher nicht bei einer *bestimmten* Anzahl von Schritten, die von der Maschine ausgeführt werden können. Dies ist nur der Fall im Hinblick auf die Faktizität der Maschine und ihrer Umgebung, und unter Heranziehung eines rigorosen Abschlusses der Maschine als unerweiterbares und singuläres System ohne Netzwerkeinbindung. Die Grenze dieses für reale Maschinen Möglichen verläuft vielmehr zwischen dem Berechenbaren und dem Unberechenbaren, wie es die Turingmaschine definiert. Das heißt, die *universale Turingmaschine* ist zwar ein theoretisches Konstrukt, aber eines, das von den Möglichkeiten aller bisher real gebauten, endlichen Maschinen gedacht werden muss. Die universale Turingmaschine bildet in diesem Sinne die Haut, die äußerste Grenze dessen, was mit Maschinen überhaupt möglich ist.



Die universale Turingmaschine als die äußerste Grenze dessen, was Maschinen überhaupt leisten können.

Um zu zeigen, was mit realen endlichen Maschinen operativ gemacht werden kann, braucht Turing allerdings, so könnte man sagen, eine gewisse Prise Unendlichkeit. Diese Prise ist allerdings so klein, dass es für das Konzept der Turingmaschine ausreicht, statt von ihrer Unendlichkeit, von ihrer *Unbegrenztheit* zu sprechen, was in diesem Kontext einen kleinen, aber feinen Unterschied darstellt. Auch mit einen unbegrenzten Band und unbegrenzter Zeit ist die Turingmaschine exakt zu dem in der Lage, was Turing als das Berechenbare definierte.

Unabhängig davon, ob sich die Konzepte der Hypercomputation als reine Papiermaschinen erweisen werden oder nicht, klafft zwischen ihnen und den realen endlichen Maschinen eine bis heute empirisch fundierte Differenz. Die universale Turingmaschine dagegen steht, weil sie die Grenze und damit, wie im nächsten Kapitel (5.2) gezeigt wird, im aristotelischen Sinne auch den Ort dessen bildet, was operative Maschinen überhaupt leisten können,[216] fest auf Seite der endlichen Maschinen. Das bedeutet auch, dass sie weniger als die Leistung eines einzelnen Mathematikers zu verstehen ist, dem sie auf einer Wiese bei Grantchester liegend[217] als höhere Fügung zugetan wurde, sondern als Ausdruck und Folge des Umgangs mit Maschinen und der dabei gewonnenen Einsichten, das heißt, eines doppelten *Begreifens* von realen Maschinen, wie es sich in Gödels Lesart dieser Maschine äußert.[218]

Die Betrachtung der Konzepte aus dem Bereich des Hypercomputation liefert noch ein weiteres Argument dafür, dass die universale Turingmaschine im oben genannten Sinne hautnah an realen operativen Maschinen verortet werden muss. Der Grund ist, dass das logische Zeitverhalten der Turingmaschine nicht undifferenziert bleiben kann.
Tatsächlich erwähnt Turing ebenso wenig wie Post[219] in seiner Beschreibung den Faktor Zeit. So schreibt auch Copeland: „[…] [T]hey said nothing about the *duration* of each primitive operation. Temporal considerations are not relevant to the functioning of the devices as described, nor to the soundness of the proofs that Turing gave concerning them.“[220] Der zweite Satz in diesem Zitat Copelands ist allerdings zu stark. Da eine unendlich schnelle Arbeitsweise der Turingmaschine die Argumentation Turings bezüglich des *Satisfactoriness Problems* destruiert – und ebenso die Argumentation bezüglich des Halteproblems, auch wenn diese strenggenommen nicht von Turing selbst stammt (vgl. Kapitel 2.2.2) – sind temporale Überlegungen für die Funktionsweise der Turingmaschine eben doch relevant. „Die Turing-Maschine selbst ist damit nicht mehr schlicht Teil (oder gar Vollendung) einer Geschichte der Medien, sondern erweist sich als das Modell einer Zeitlichkeit, die bislang […] verkannt wurde.“[221]

[216] Darin, dass die Turingmaschine die Grenze bzw. der Ort dessen ist, was operative Maschinen überhaupt leisten können, liegt ein Ansatzpunkt, um Gödels Formulierung der „machine with a finite number of parts“ in ein deutlicheres Licht zu rücken.
[217] Vgl. Hodges 1983, 113.
[218] In die soeben explizierten Gedankengänge fällt auch die Differenz zwischen Schema und tatsächlicher operativer Ausführung. Dass sich eine endliche Tabelle aufstellen lässt, mittels derer π in voller Länge ausgegeben werden kann, heißt eben nicht, dass π auch tatsächlich ausgegeben werden kann. Ein Schema aufzustellen ist daher nicht gleichbedeutend damit, etwas tatsächlich operativ ausgeführt zu haben.
[219] Vgl. Post 1936.
[220] Copeland 2002, 282.
[221] Ernst 2012a, 74.

Gerade um den Ort aller operativen Maschinen bilden zu können, muss die universale Turingmaschine ein theoretisches Konstrukt bleiben, indem sie eine Prise Unendlichkeit bzw. Unbegrenztheit von der Art enthält, für die sich, ganz im Gegensatz zu den unendlich schnellen oder unendlich präzisen Maschinen der Hypercomputation, keine auch nur im Entferntesten realisierbaren Konzepte finden lassen. Die universale Turingmaschine liegt damit gerade nur deswegen rein im Raum des *Symbolischen,* weil sie die äußerste Grenze des im *Realen* maschinen- und medientechnisch überhaupt Möglichen absteckt. Für sie kann damit der bekannte Satz Lichtenbergs Geltung verlangen: „Auf der Grenze liegen immer die seltsamsten Geschöpfe."[222]

4.3.3 Erfassung

Nur unendliche Folgen von Symbolen können unberechenbar sein. Das heißt, alles, was sich in endlichen Folgen von Symbolen ausdrücken lässt, kann von einer Turingmaschine berechnet werden. Die Frage, ob sich ein Modell für den 3D-Druck, für die synthetische Biologie, für die Bewegung von Industrierobotern usw. erstellen lässt, das von einer Maschine nicht berechnet werden kann, ist daher müßig, solange es sich bei allen diesen Modellen um solche von *endlicher Größe* handelt. Die Nutzung von Maschinen führt in dieser Hinsicht nicht zu einem Verlust an möglichen Strukturen oder eines Zugangs zu unberechenbaren Teilen der »Welt«[223]. Es werden wohl auch keine Fehler zu erwarten sein, die durch diese Art des technischen Umgangs mit »Welt« intern, das heißt in den Rechnungen selbst, provoziert werden.[224]

Alles Fragen nach der Adäquatheit der Bearbeitung von »Welt« mittels digitaler Medien lässt sich also – zumindest aus Sicht der Berechenbarkeit – auf die Frage nach der *Erfassung* von Daten reduzieren. Dies mag den Anschein erwecken, als sei die Untersuchung digitaler Maschinen darüber hinaus überflüssig. Im Schlusskapitel dieser Arbeit wird allerdings gezeigt werden, dass eine genaue Betrachtung der Turingmaschine bzw. der mechanischen Prozedur es ermöglicht, Erfassung unter geändertem Vorzeichen zu betrachten: Anstatt von einer implizit oder explizit als so oder so aufgefassten sogenannten realen »Welt« auszugehen und zu überlegen, wie dieser in Daten entsprochen werden kann, wird hier umgekehrt von den technischen Bedingungen des digitalen Raums ausgegangen, der diese Daten *erwartet*.

---

[222] Lichtenberg 1990, 45.

[223] »Welt« hier im Sinne Heideggers mit Anführungszeichen, vgl. Heidegger 1927, 63-66. Um philosophische Missverständnisse zu vermeiden, wird Heideggers Unterscheidung im weiteren Verlauf durchgehalten.

[224] Vgl. zu dieser Frage das großartige Gespräch zwischen dem jungen Turing und Wittgenstein, zitiert in Hodges 1983, 180.

5. Berechenbarkeit als Sphäre

Eine der wesentlichen Fragen einer halbtechnischen Medienwissenschaft – wenn nicht die Gretchenfrage selbst – ist die nach dem Verhältnis von Maschine und Medium. Um dieses hier bisher unreflektierte Verhältnis zu explizieren, werden im Folgenden die Standpunkte zweier Medientheoretiker des Programms einer Berliner Medienwissenschaft verdeutlicht. Vor diesem Hintergrund wird dann das Konzept einer Sphäre der Berechenbarkeit dargestellt.

5.1 Maschine vs. Medium

Schon 1986, in Grammophon – Film – Typewriter, legt Friedrich Kittler dar, dass der Computer nicht nur ein Medium ist, sondern ein Medium dazumal, das alle anderen Medien aufnehmen und übernehmen wird: „Und wenn die Verkabelung bislang getrennte Datenflüsse alle auf eine digital standardisierte Zahlenfolge bringt, kann jedes Medium in jedes andere übergehen. […] [E]in totaler Medienverbund auf Digitalbasis wird den Begriff Medium selber kassieren.“[225] Zwei Jahre später lässt Kittler dieser Aussage noch einmal klare Worte folgen:

> Die Entwicklung technischer Medien, wie sie vom digitalen Übertragungsmedium Telegraphie über die analogen Speichermedien Schallplatte und Film zu ihren Übertragungsmedien Radio und Fernsehen führte, kommt an ein logisch perfektes Ende. Alle anderen Medien sind in die Diskrete Universale Maschine grundsätzlich überführbar.[226]

Dieses Bild des Aufgehens aller althergebrachten Medien im universalen Medium Computer ist eines der grundlegenden Motive in Kittlers Œuvre: Die „[…] Computertechnik in ihrer Macht, alle anderen Medien zu integrieren“[227], entfaltet sich auf der Tatsache, dass „[…] jeder Computer alle anderen Computer imitieren […]“[228] kann. „Erst Entwicklungen der letzten zwei Jahrzehnte haben den Computer zu einem Medium gemacht, das im Prinzip und wohl auch schon in nächster Zukunft alle anderen Medien wird sein können […], [so] daß Computer zu universalen Medien werden […]“[229] und anfangen, „[…] alle anderen Medien zu fressen, das Telefon, den Telegraf und das Faxgerät und bald auch das Bild und den Ton und die CD“[230], wie auch „[…] Filme, Schallplatten, Radios […] heute dabei sind, eins nach dem anderen im Universalmedium Computer zu landen.“[231]

---

[225] Kittler 1986, 8.
[226] Kittler 1988, 189f.
[227] Kittler 1995a, 174.
[228] Ebd., 170.
[229] Kittler 1995c, 113.
[230] Kittler 1995b, 135.
[231] Kittler 1996b, 164.

Klar ist: Der Computer ist – wie zu erwarten – in Kittlers Augen ein Medium und dieses Medium ist als universales Medium dabei, alle anderen Medien zu „fressen". Warum dieses totale Medium[232] Computer den Medienbegriff dabei „selber kassieren" wird, ist auf Basis der bisher versammelten Zitate noch nicht klar zu erkennen.

Für Kittler ist der Computer aber nicht nur ein Medium, er ist auch eine Maschine. Theoretisch kann der Computer als Maschine laut ihm auch den sogenannten Menschen imitieren: „Diese Papiermaschine nämlich tut genau das, was Mathematiker tun, wenn sie Zahlen und Buchstaben auf einem Stück Papier manipulieren, nur eben von allein."[233] Dass diese anthropozentrische Auslegung auch im Sinne Turings richtig ist, fand bereits Erwähnung (vgl. Kapitel 2.1), soll aber im Rahmen dieser Arbeit nicht weiter interessieren. Laut Kittler imitiert die Turingmaschine jedoch nicht nur Mathematiker. Sie ist darüber hinaus auch die „Maschine, die nach Turings Definition alle anderen Maschinen soll imitieren können […]"[234], denn: „Laut Turing sind universale Maschinen eben dadurch definiert, daß sie die Operationen beliebiger anderer Maschinen speichern und lesen, also deren Programm ausführen können, wobei der Unterschied zwischen Ausführung und Imitation hinfällig […] wird."[235]

Dass eine Maschine sich in einer anderen implementiert findet, kann also heißen, dass eine Maschine die andere *imitiert*. In einem stärkeren Sinne darf dies laut Kittler aber auch so verstanden werden, dass die universale Maschine jede andere Maschine *sein kann*:[236] „1936 jedoch gab Alan Turing […] eine Maschine an, die alle anderen Maschinen, deren Beschreibung sie eingelesen hat, selbst sein kann."[237] Die Turingmaschine ist also nicht nur *conditio sine qua non* für Turings Imitationsspiel, zumindest, solange es um maschinelle Teilnehmer geht,[238] sondern auch die Maschine, „[…] die ihrem Erfinder zufolge alle anderen Maschinen sein kann […]."[239]

---

[232] Vgl. zu dieser „totalen Maschine" als Folge eines „totalen Kriegs" und der Eroberung eines „totalen Kommunikationssystems" Hodges 1983, 389.

[233] Kittler 1996a, 112, vgl. auch Kittler 1995b, 159.

[234] Kittler 1996a, 127f, vgl. auch 132 und Kittler 1995b, 159.

[235] Kittler 1997, 63. So beruhen „[…] alle Eskalationen der Mediengeschichte auf dem elementaren Sachverhalt, daß sämtliche Operationen einer Maschine A von einer anderen Maschine B imitiert werden können" (Kittler 2000, 260), wobei gilt: „Universale Turingmaschinen brauchen nur mit der Beschreibung (dem Programm) einer beliebigen anderen Maschine gefüttert zu werden, um diese Maschine effektiv zu imitieren." (Kittler 1993, 287).

[236] Es wäre von einigem Interesse, Kittlers Maschine, die alle Maschinen „sein kann", mit Pias' Beschreibung des Smartphones, das für ihn „ein Telefon, ein Fotoapparat, ein Walkman, […] und alles mögliche andere *ist*" (Pias 2015, 34), zu vergleichen. Kittlers Formulierung spannt einen Möglichkeitsraum auf: Es ist eben ein Unterschied, ob etwas *X*, *Y*, *Z ist*, oder ob es *X*, *Y*, *Z sein kann*.

[237] Kittler 2000, 260. Vgl. auch ebd., 262: „Zwei Jahrhunderte systematischer Eskalation sind auf eine Maschine zugelaufen, die alle anderen Maschinen sein kann."

[238] Vgl. Turing 1950.

[239] Kittler 1995c, 112f. Vgl. auch Kittler 1996b, 167; „[…] die Universale Diskrete Maschine, die ja alle anderen Maschinen sein kann […]."

Durch Kittlers Œuvre – zumindest durch die Arbeiten aus dem zweiten der drei Stadien der intellektuellen Karriere Kittlers, seiner Betätigung als Medienwissenschaftler[240] – ziehen sich diese beiden Auffassungen der Turingmaschine bzw. des Computers: Der Computer ist zum einen das Medium aller Medien, weil er als universales Medium alle anderen Medien absorbiert, zugleich ist der Computer aber auch die Maschine aller Maschinen, weil er alle anderen Maschinen imitieren, wenn nicht sogar sein kann. Kittler postuliert hierbei nicht nur das Aufgehen aller Einzelmedien im digitalen Medienverbund, sondern eben auch den Zusammenfall von Medium und Maschine. Ein Zusammenfall zum Nachteil des Mediums dazumal, da dieser Vorgang den „Begriff Medium selber kassieren"[241] wird. Es können nun für dieses Ende des Begriffs Medium bei Kittler zwei Lesarten angeboten werden, was einen kurzen Einschub erforderlich macht:

Nach der ersten Lesart reicht es für das Verschwinden des Begriffs Medium aus, dass die einzelnen Medien in einem universalen Medium aufgehen. Tatsächlich drücken die versammelten Zitate zum Medium aller Medien auch genau dies aus und nicht mehr, so dass man dieser ersten Lesart folgen sollte, wenn da nicht die mit ihr verbundene Schwierigkeit wäre, zu begründen, warum dann nicht zumindest das totale Medium Computer selbst noch ein dem Begriff würdiges Medium sein sollte. Schließlich war der Computer doch zuvor explizit als Medium definiert worden und erfüllt gerade nach der Aufnahme aller anderen Medien immer noch die Kittlersche Trias vom Übertragen, Speichern und Verarbeiten.[242] In der Überzeugung, dass Kittler, wenn er trotz dieser Schwierigkeit die erste Lesart verfolgt hätte, den Plural Medien verwendet hätte – um anzuzeigen, dass es ihm lediglich um das Verschwinden der Vielfalt der Medien ginge – wird hier, zumindest solange sich keine schlüssigere als sie finden lässt, der zweiten Lesart der Vorzug gegeben:

Diese zweite Lesart beruht auf dem Umstand, dass der Computer für Kittler zugleich Medium und Maschine ist. Kassiert wird der Begriff des Mediums laut dieser Lesart bei der Kreuzung von Medium und Maschine, wie sie in den Computern stattfand und stattfindet. Eine Kreuzung, bei der sich das Medium gegenüber der Maschine – aus zugegebenermaßen anhand der gegebenen Zitate nicht erkennbaren Gründen – als rezessiv erweist. Zwar ist es laut Kittlers Wort immer noch ein „Medienverbund",[243] der den Begriff des Mediums kassiert. Auch aus diesem Blickwinkel aber sind es die digitalen Maschinen und deren Imitations- und Netzwerkfähigkeiten, die an diesem Verlust ursächlich beteiligt sind.

---

[240] Wenn man der diskretisierenden Einteilung Winthrop-Youngs folgt. Vgl. Winthrop-Young 2005, 17.
[241] Kittler 1986, 8.
[242] Vgl. dazu auch den von Kittler mitherausgegebenen Band *Computer als Medium* (Bolz/Kittler/Tholen 1999).
[243] Kittler 1986, 8.

Man könnte nun unterstellen, dass die Annahme einer Sphäre der Berechenbarkeit als gemeinsame Grenze von Maschine und Medium ein ähnliches Zusammenfallen dieser beiden Begriffe impliziere. Eine Reduktion dieser Art hieße aber, einen begrifflichen Verlust hinzunehmen und die Beiträge aller der Ansätze in den Wind zu schlagen, die es unternehmen, Maschine und Medium begrifflich voneinander getrennt zu denken.
So formuliert Ernst auf Basis des Zeitverhaltens von Maschinen und Medien eine klare Unterscheidung zwischen klassischen Maschinen und elektronischen Medien. Ernst gibt dabei eine Definition von Medium, die den Computer mit einschließt:

> By definition, Greek *metaxy* […] and its Latin translation *medium* is grammatologically […] and technically located inbetween beginning and end, sender and receiver, data input and output, and so forth: as the intermediary, thus temporally ephemeral channel of transmission […], or as the processing unit in computing.[244]

Gleichzeitig betont Ernst aber die Prozessualität „hochtechnischer, *zeitgebender* Medien"[245] und verweist entsprechend auf den „[…] Unterschied von Maschinenzeit und elektronischer Medienzeit […]."[246] Transklassische Medienmaschinen unterscheiden sich demnach von den klassischen Maschinen durch ihre zeitliche Existenzweise:[247]

> [Die] Kybernetik [modelliert] (in Nachfolge Hermann von Helmholtz') den Menschen auf physiologischer und neuronaler Ebene […] – gerade nicht mehr nach dem Modell der gleichmäßig getakteten Maschine, sondern der zeitkritischen und rückkopplungsfähigen Signal- als Informationsverarbeitung. […] Neben evolutionären Spuren kommen hier kleinste Zeitmomente ins Spiel, wie sie im Elektronischen vollends zum Zuge kommen. Hier gilt es nicht mehr, schlicht Maschinen zu denken, sondern elektromathematische Medien.[248]

Wo für Kittler der Begriff Medium auf- und verlorengeht, differenziert Ernst auf Basis des Zeitverhaltens die Begriffe Maschine und Medium aus. Dabei ergibt sich ein komplexeres Bild, das die Unterschiede zwischen Maschine und Medium aufscheinen lässt, andererseits aber auch Kittlers Diagnose vom Ende der Mediengeschichte mit aufnimmt:

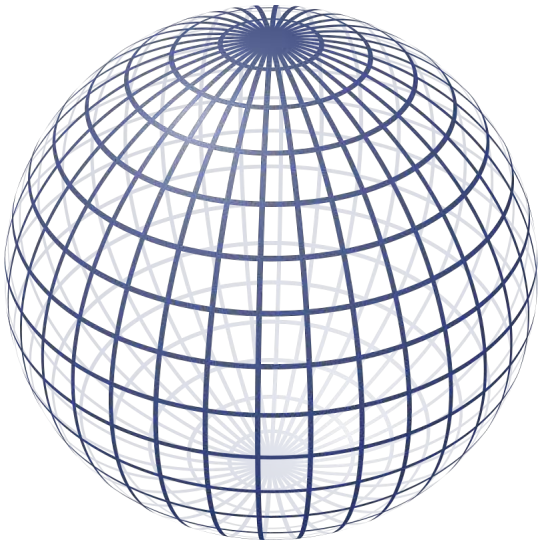
> Historisch [an klassischen Dokumenten] sind die materiellen Kontexte, das alte Manuskriptpapier, die Entropie der Tinte auf biochemischer Ebene; auf der symbolischen (also Informations-)Ebene aber sind solche Dokumente unhistorisch, insofern sie die lineare Zeitlichkeit durch eine andere, nicht-physikalische Zeitverfaßtheit unterlaufen. In der symbolprozessierenden Maschine namens Computer kreuzen sich am Ende beide Zeitweisen – was Mediengeschichte ebenso auskreuzt wie durchscheinen läßt.[249]

---

[244] Ernst 2009b, o. S., im Kapitel „Algorithmic time". Vgl. dazu beispielsweise die Definition von Computer im weitesten Sinne: „[I]t is rational to adopt the view that any physical system capable of providing reliable output is a potential computing device." (Yao 2002, 3)
[245] Ernst 2009a, 32.
[246] Ernst 2012a, 92.
[247] Vgl. Ernst 2012b, 350.
[248] Ernst 2012c, 115f.
[249] Ernst 2012b, 360.

5.2 Das Konzept der Sphäre

Ist es denkbar, dass ein Ansatz wie der hier verfolgte, der eine gemeinsame Grenze von Medien und Maschinen anzunehmen bereit ist, implizit auch von einem Zusammenfall dieser beiden Begriffe ausgehen muss? Anstatt in diesem Kapitel einer bestimmten Sichtweise das Wort zu reden, wird gezeigt, dass das Konzept der Berechenbarkeit als Sphäre digitaler Medien entwickelt werden kann, ohne über das Verhältnis von Medien und Maschine und deren etwaiges Zusammenfallen in den digitalen Medienverbünden unserer Tage ein Urteil fällen zu müssen. Begründet liegt dieser Umstand im Begriff der Sphäre selbst. Mathematisch gesehen nämlich bezeichnet dieser Begriff nicht eine Kugel und deren Inhalt, sondern lediglich die *Oberfläche* einer Kugel, die Grenzfläche also, die diese Kugel definiert.

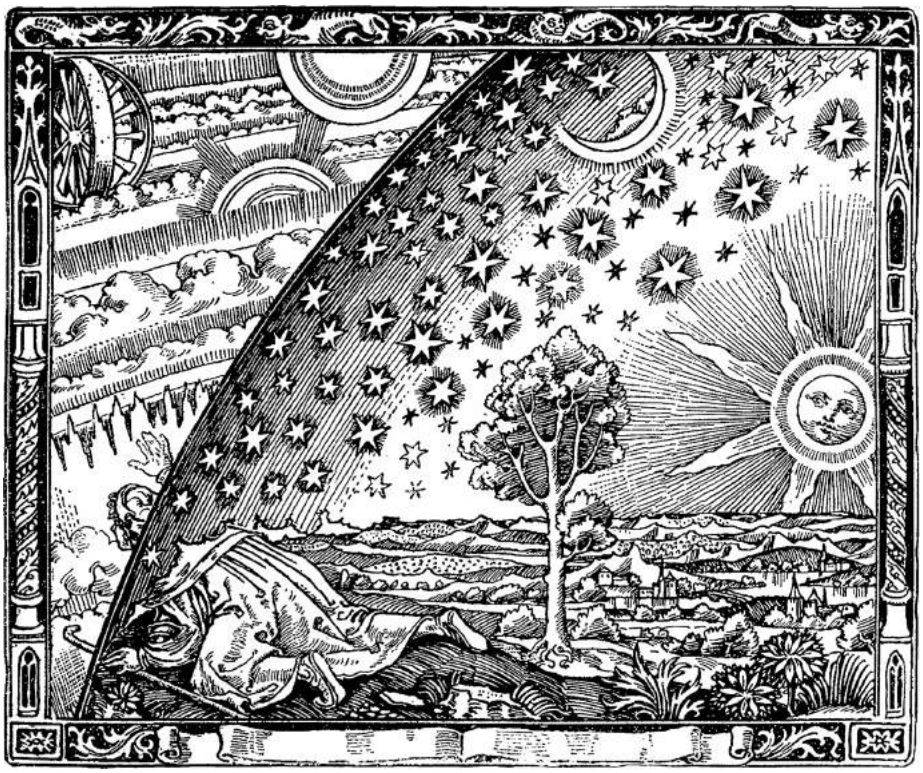


Darstellung der mathematischen 2-Sphäre[250]

Die Sphäre ist also mit dem gleichzusetzen, was in der Übersetzung von Aristoteles' *Physik* Buch IV die „Volldefinition des Ortes" heißt: „Der Ort ist die unmittelbare (d. h. nächstgelegene) nicht in Bewegung begriffene Angrenzungsfläche des (den Gegenstand) umschließenden Körpers."[251] Eine Hülle oder Grenze also, und nicht das Innere selbst. Ganz bewusst ruft die Verwendung des Begriffs Sphäre – im Altgriechischen σφαίρα, d. h. eben Hülle oder Ball – damit auch die altertümliche und mittelalterliche Bedeutung von Sphäre auf: Die Sphäre als Himmelsgewölbe, als kristallene Hohlkugel, auf der die Gestirne sitzen.

[250] Abbildung von Wikipedia, Artikel zur mathematischen Sphäre. Download am 20.5.16 von: https://de.wikipedia.org/wiki/Sphäre_(Mathematik)#/media/File:Sphere_wireframe_10deg_6r.svg
[251] Aristoteles *Physik* IV, 212 a.

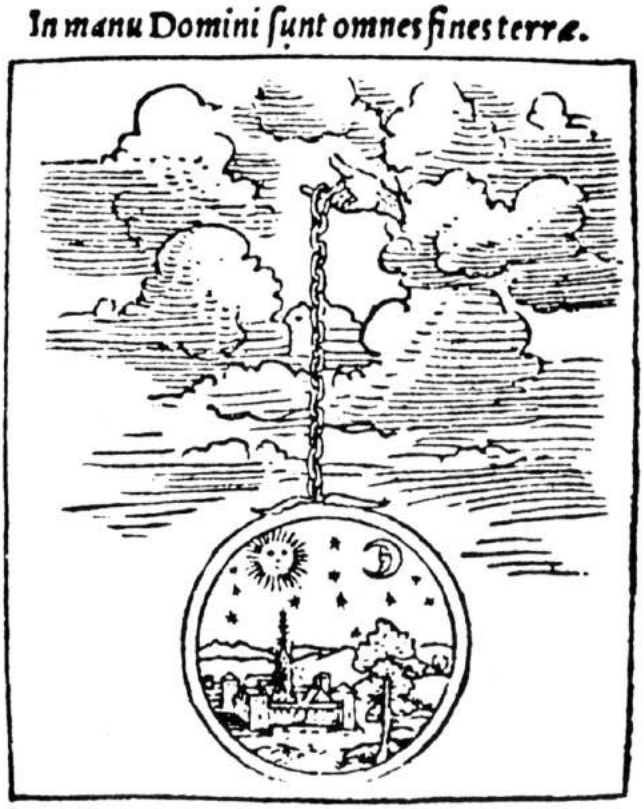
In diesem Sinne formt die Sphäre der Berechenbarkeit als Sphäre hochtechnischer Medien die Grenzfläche dessen aus, was nach heutigem Ermessen mit realen Medien und Maschinen höchstens möglich sein kann – also exakt das, was von der universalen Turingmaschine gezeigt werden konnte (vgl. Kapitel 4.3.2). Es ist mithin die universale Turingmaschine selbst, die diese Sphäre der Maschinen definiert und es ist Turings Leistung, diesem Zusammenhang durch die Wahl eines Maschinenmodells Rechnung getragen zu haben. Dass hochtechnische Medien und Maschinen aus Sicht der Berechenbarkeit gleiche Extension haben, kann als technisch fundierter Ansatzpunkt für eine Befragung des Verhältnisses von Maschine und Medium dienen. Ob und unter welchen Bedingungen Maschine und Medium zusammenfallen oder immer schon zusammengefallen sind, ist aber eine Frage nach dem Inhalt der Sphäre, nicht nach der Sphäre selbst als der gemeinsamen Oberfläche, und darum auf Basis des Konzepts der Sphäre vorerst nicht angebracht.



Flammarions Holzstich: Erschienen auf Seite 163 in Nicolas Camille Flammarions *L'Atmosphère. Météorologie popu-laire* von 1888. Das Bild ist untertitelt: „Un missionnaire du moyen âge raconte qu'il avait trouvé le point où le ciel et la Terre se touchent… [Ein Missionar des Mittelalters erzählt, dass er den Punkt gefunden habe, wo der Himmel und die Erde sich berühren…]."[252]

[252] Abbildung von Wikipedia, Artikel zu Camille Flammarion. Download am 19.5.16 von: https://upload.wikimedia.org/wikipedia/commons/8/87/Flammarion.jpg

Es zeigen sich hier zwei Extreme, zwischen denen zu vermitteln ist: Zum einen ist es möglich, Berechenbarkeit zum zentralen Begriff einer technischen *philosophia perennis* zu machen, und zu behaupten, die Entwicklung der Welt wäre zum ersten und endgültigen Mal richtig verstanden, da sie jetzt aus diesem Blickwinkel betrachtet wird. Zum anderen ließe sich Berechenbarkeit auch als ein kontingentes, genealogisch zu verstehendes und damit veränderliches und vergängliches Ordnungssystem verstehen. Eine Taxonomie, die nichtsdestoweniger das Ordnungssystem dieser Zeit ist und bleiben wird, solange technische Medien und Maschinen diese noch-menschliche Lebenswelt prägen, und solange keine Alternative zu mechanischen Prozeduren und damit zur Berechenbarkeit aufgekommen ist.



Sphärenbild, Gottes Allmacht verherrlichend, 1508.[253]

In diesem Sinne bildet die kristallene Himmelssphäre der Berechenbarkeit eine technisch bislang unüberbietbare Grenze für alles das, was maschinen- und menschenmöglich ist. Jenseits dieser Sphäre liegt damit buchstäblich eine Metaphysik der Maschine und der hochtechnischen Medien, die wir symbolisch erschließen, aber wohl nicht *operativ* machen können. Wie der Inhalt dieser Sphäre als »Welt« beschaffen sein mag, ist unerheblich für die Tatsache, dass das Äußere dieser Sphäre der eigentlichen Erschließung immer noch harrt.

[253] Abbildung aus Galilei 1613, 171.

6. Die Erdung der Sphäre

Das Konzept der Berechenbarkeit als Sphäre kann wie gezeigt als inhaltlich offenes Modell expliziert werden. Dennoch soll im Abschluss der Arbeit eine mögliche Erdung versucht und zumindest einige der mit dem Konzept einer Sphäre der Berechenbarkeit verknüpften Implikationen und Gedanken adressiert werden.

So existiert gemeinhin eine strikte Trennung zwischen Mechanischem und Elektrischem bzw. Elektromagnetischem. Die Mechanik der klassischen Physik – mit den Unterdisziplinen Statik und Dynamik, wie sie bei Lagrange entwickelt wurde,[254] oder der modernen Aufteilung in Kinematik und Dynamik – steht der Elektrodynamik als eigenständiger und klar zu unterscheidender Bereich gegenüber. Diese Trennung wird auch in anderen Disziplinen kaum unterlaufen. So unterscheidet beispielsweise Khan für natürlich hervorgebrachte Klänge zwischen dem „Aeolian […] [as] a mechanical music in that the actions of the wind, vibrating strings, and the resulting sound are of the same class of mechanics" und dem „Aelectrosonic [that] moves from electromagnetism to sound, that is, from the class of electromagnetic energy to the class of mechanical energy."[255] Ebenso werden in der Medizin Behandlungsmethoden in mechanische Verfahren, Thermotherapie, Elektrotherapie und Strahlentherapie eingeteilt, auch wenn sich die exakten Bezeichnungen und Einteilungen mit der Entwicklung der Methoden wandeln. Dabei findet sich sogar explizit die Rede von mechanischen Prozeduren.[256] Auch Rechengeräte selbst werden auf diese Weise unterschieden. So trägt Zuses Arbeit ‚Entwicklungslinien einer Rechengeräte-Entwicklung von der Mechanik zur Elektronik' dieser Differenz schon im Titel Rechnung.[257] Dass die Unterscheidung zwischen Mechanischem und Elektromagnetischem für Ernst die Differenz zwischen klassischen Maschinen und Medienmaschinen mitmarkiert, wurde bereits in der Differenz zu Kittlers Postulat vom Verschwinden des Mediums in der Maschine behandelt:

> Gerade im Erstaunen [Barkhausens] über die Analogie [zwischen mechanischem Pendel und elektromagnetischer Schwingungserzeugung] liegt das Wissen (oder die Ahnung) um die abgrundtiefe epistemische Differenz: hochtechnische Mediengefüge *versus* klassische Maschinen.[258]

Hier zeigt sich klar, dass das Mechanische und das Elektromagnetische für Ernst epistemologisch gesehen verschiedenen Bereichen angehören. Diese Kluft reicht dabei so tief, dass sie nicht länger narrativ zu einer Fortschrittsgeschichte gekittet werden kann.

---

[254] Vgl. Lagrange 1788.
[255] Khan 2013, 7.
[256] Bakker et al. 1932, 674-691.
[257] Vgl. Zuse 1962.
[258] Ernst 2012a, 240.

Ähnlich stark, oder gar noch stärker, hat zuvor auch McLuhan in *Understanding Media* die Kluft zwischen mechanischer und elektrischer Ära gezeichnet.[259] So schreibt er dort: „Automation ist not an extension of the mechanical principles of fragmentation and separation of operations. It is rather the invasion of the mechanical world by the instantaneous character of electricity.“[260] Für McLuhan ist es diese Instantanität der Elektrizität, die sie von allen mechanischen Apparaturen abhebt und die sie dazu befähigt, das mechanische Zeitalter zu beenden, das mit Gutenberg vollends in den Tritt gekommen war.[261] Dabei liegt auch hier kein fließender Übergang von einer Ära in die andere vor, sondern ein harter Bruch, der schon zu Beginn von *The Medium is the Massage* in aller Klarheit formuliert wird:

> The medium, or process, of our time – electric technology – is reshaping and restructuring patterns of social interdependence and every aspect of our personal life. It is forcing us to reconsider and reevaluate practically every thought, every action, and every institution formerly taken for granted.[262]

Auch hier zeigt sich, dass der instantane Charakter der Elektrizität für McLuhan kaum genug betont werden kann: „Electric circuitry has overthrown the regime of "time" and "space" and pours upon us instantly and continuously the concerns of all other men.“[263]

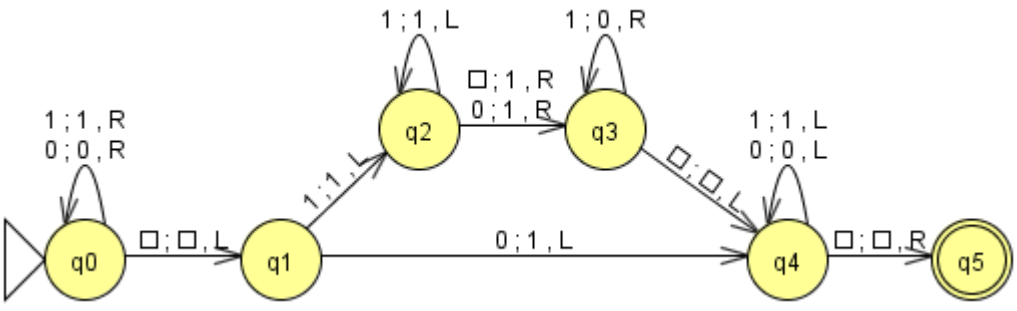
Raoul Dufy: *La Fée Electricité*. 1937, Wandbild, 600 Quadratmeter. Ausschnitt: Linke Bildhälfte.[264] „[…] Darstellung des modernen, lichtüberfluteten Alltags, über den eine Fee, die »Fee Elektrizität«, hinwegfegt. Sie steht für den Zauber, den das unsichtbare, abstrakte Phänomen der Elektrizität bewirkt.“[265]

[259] Vgl. dazu McLuhan 1964, 25f.
[260] Ebd., 349.
[261] Vgl. ebd.
[262] McLuhan/Fiore 1967, 8.
[263] Ebd., 16.
[264] Abbildung von Bianic 2013 (Internetquelle).
[265] Braun/Kaiser 1990, 272.

Die instantane Elektrizität findet sich ebenso ein Jahr später in *War and Peace in the Global Village*: „It is not only our educational system that cannot stand up to the new electronic speed of information movement. The stock exchanges of the world are just as helpless and will disappear under the impact of the computer in a few years.“[266] McLuhan neigt in dieser Arbeit etwas zur Überzeichnung, der Tenor jedoch ist unverkennbar der selbe: Es ist die Elektrizität, die das neue Zeitalter durch ihre instantane Wirkung hervorruft und definiert. Wie ist es vor dem Hintergrund dieser „epistemischen Differenz“ möglich, dass eine *mechanische* Prozedur eine Sphäre für *elektronische* Medien aufspannt?
Bezeichnend ist, dass das Erklärungsmodell von der instantanen Elektrizität im letzten Kapitel von *Understanding Media*, in dem es McLuhan um Automation geht, mit einem Mal deplatziert wirkt. So schreibt Frey mit explizitem Bezug auf dieses Kapitel:

> Sicher ist es richtig, dass die Entwicklung der elektronisch gesteuerten Bauteile in Automaten enorm dazu beigetragen hat, die Leistung von Automaten zu erhöhen. Aktuell ist Elektrizität unabdingbar, um Automaten zu betreiben. Doch diese Energieform ist den Automaten nur äußerlich, sie ist nicht tatsächlicher Bestandteil der Automaten und Computer. Der textgesteuerte Automat benötigt, um zu arbeiten, zwar Energie jedoch nicht originär Elektrizität. Dies unterscheidet ihn zum Beispiel von einem Fernsehen, das der Elektrizität vom Grunde her bedarf. Die elementaren Arbeitsvorgänge von Automaten setzen anders als die Medien Telefon, Radio oder Fernsehen Elektrizität nicht *vom Prinzip her* voraus.[267]

Was für einige der Medien im Plural wie Fernsehen, Telefon und Radio noch notwendige Bedingung war, ist für das Medium aller Medien nur ein – wenn auch besonders geeignetes – Milieu unter vielen. Die Universalität der digitalen Rechner liegt daher keineswegs in der Instantanität und Vielseitigkeit[268] der Elektrizität begründet. Das anthropozentrische und idealisierte Bild von der Fee Elektrizität weicht so dem Zustandsdiagramm einer Turingmaschine, die auf alle möglichen Arten umgesetzt werden könnte.



Zustandsdiagramm einer Turingmaschine. Wie leicht nachzuvollziehen ist, erhöht diese Maschine eine beliebige Binärzahl auf dem Band um 1. Interessanterweise würde ein leeres Band zum Absturz führen.[269]

[266] McLuhan/Fiore 1968, 183.
[267] Frey 2009, 14.
[268] Vgl. dazu McLuhan 1964, 350.
[269] Abbildung von Becker 2014 (Internetquelle).

Diese Erkenntnis ist nun spätestens seit Kittler nichts Neues. Allerdings geht Kittler in diesem Punkt nicht weit genug. Die Turingmaschine ist eben nicht nur eine aufs Prinzip abgemagerte Schreibmaschine,[270] sondern das Prinzip des Mechanischen selbst. Das heißt, Mechanisierung bezieht sich aus dieser Perspektive *nicht nur* auf das mechanische „Gestänge und Geschiebe und Gerüste“[271] klassischer Maschinen. Mechanisiert werden kann auch mittels elektrischer, elektromagnetischer, biologischer, genetischer und überhaupt aller Vorrichtungen, die auf eine bestimmte Eingabe mit einer hinreichend verlässlichen Ausgabe antworten. Die tatsächliche Umsetzung ist für die prinzipiellen Grenzen der Berechenbarkeit unerheblich. Es geht um die Mechanisierung als Sinn- und Denkentleerung – zumindest, was das Denken im klassisch humanistischen Sinne betrifft. Dabei geht es nicht nur um „mechanized mathematics,“[272] wie es unter anderem schon bei Mill[273] und Boole[274] noch vor dem Aufstieg der formalisierten Mathematik verhandelt wurde. Auf dem Spiel steht viel mehr, nämlich welche Sprachen,[275] Tätigkeiten und Gedanken, d. h. welche Aspekte der »Welt« in diesem Sinne überhaupt bis zu welchem Grad mechanisiert werden können.[276] Dass der so benannte Vorgang natürlich das oft und in vielerlei Auslegungen beschworene Programm der Mechanisierung auf den Plan ruft, das zu Beginn der Neuzeit bei Descartes eine klare Ausformulierung fand, darf nicht darüber hinwegtäuschen, dass das Mechanische historisch kein stabiler Begriff ist, sondern an sich schon vor ein „terminologisches Problem“[277] stellt. Es geht also nicht darum, das Mechanische als einen festen Begriff über die Turingmaschine zu stülpen, sondern umgekehrt, an Turings Definition – vor allem an der Lesart Gödels – das Mechanische in dieser Ausprägung erst abzulesen. Das zugrundeliegende mechanische Prinzip, wie es sich hier zeigt, mag für Turing vielleicht tatsächlich an einer Schreibmaschine erkennbar gewesen sein. Es aber auf seine universale und wohldefinierte Form gebracht zu haben, bleibt seine außerordentliche Leistung. Inwieweit dieses Prinzip allerdings rückübersetzt werden darf und auch für die μηχανή der antiken Griechen und die Maschinen der Descartes und La Mettries Geltung verlangen darf, muss an dieser Stelle offen bleiben.

[270] Vgl. u. a. Kittler 1986, 31 und 1992, 263.
[271] Heidegger 1962, 20.
[272] Ernst 2009, o. S., im Kapitel „Finite State Machines and the *Halteproblem*“.
[273] Vgl. Mill 1843.
[274] Vgl. Boole 1847.
[275] Vgl. Priestley 2011, 8-15.
[276] Es hat den Anschein, dass sich grundsätzlich nichts der Mechanisierbarkeit entzieht, solange es auf irgendeine Art und Weise *messbar* und damit in Daten und Modelle übersetzbar ist. Lediglich das Bewusstsein als solches und die Qualia, deren Zusammenhang mit den Konfigurationen des Gehirns immer noch einer wissenschaftlichen Erklärung harrt, entziehen sich auf ominöse Art und Weise dem messtechnischen Zugriff und damit auch einer Mechanisierung.
[277] Dijksterhuis 1951, 481.

In diesem Sinne verkennt McLuhan das Mechanische im Digitalen. Weder ist das elektrisch betriebene Medium Computer technisch gesehen tatsächlich instantan – vor allem vor dem Hintergrund der Konzepte aus der Hypercomputation kann dies nicht mehr behauptet werden – noch ist das Elektrische die notwendige Bedingung für die Effektivität der Computer. Wenn McLuhan also wie bereits zitiert schreibt: „Automation ist not an extension of the mechanical principles of fragmentation and separation of operations. It is rather the invasion of the mechanical world by the instantaneous character of electricity“[278], so liegt er in diesem Punkt seiner sonst so korrekten Analyse falsch. Unter den hier herausgearbeiteten Prämissen scheint eher umkehrt das Mechanische das Elektrische zu unterwandern, so dass sich in Bezug auf die realen Medien dieser Zeit sagen lässt, das Digitale ist das Mechanische im Elektrischen.

Dies bedeutet aber nicht, dass bei dieser Analyse die „epistemische Differenz“ zum Elektrischen einfach vernachlässigt werden darf. Freys Vorwurf gegen McLuhan gilt nur solange, als lediglich die *Effektivität* von Maschinen, d. h. die Gesamtmenge des operativ Möglichen betrachtet wird. Sobald auch die *Effizienz* der realen Geräte, also ihr In-der-Zeit-sein mit einbezogen werden soll, muss mit Kittler wieder die Rede davon sein, dass „elektronische Medien Information sind, die das Medium Elektrizität als ihre Energie oder Umwelt immer schon voraussetzt“[279]. Wie Karl Ernst von Baer in einem wunderbaren Abschnitt seiner Rede zur Eröffnung der Russischen entomologischen Gesellschaft in Sankt Petersburg darstellte, ist es für die menschliche Welt eben über alle Maßen entscheidend, in welcher relativen Geschwindigkeit Prozesse der äußeren Umwelt ablaufen.[280] Eine Umsetzung von Computern in anderen Milieus als dem Elektrischen ist also möglich. Es bleibt aber der Umstand bestehen, dass sich das Elektrische in besonderer Weise anbietet, um die Geschwindigkeit in der Maschine umzusetzen, die allein schon für die Täuschung der Wahrnehmung des sogenannten Users notwendig ist.[281] Dies zeigt allerdings, dass es sich bei Fragen der Effizienz schnell um ein anthropozentrisches Fragen handelt. So scheint auch McLuhans Kapitel zur Automation mehr soziologisch ausgerichtet als technisch fundiert.

Dieses digitale Zeitalter als ein mechanisches bezeichnen zu wollen, ist nicht nur mit Blick auf die Historie sicherlich unzulässig. Vielmehr ist es die Kopplung aus mechanischem Prinzip, d. h. der mechanischen Prozedur, wie sie von der Turingmaschine definiert wird, und dem faktischen Schalten und Walten der Elektrizität, das die digitalen Medien dieser technisierten Zeit grundlegend kennzeichnet – als *das Mechanische im Elektrischen.*

---

[278] McLuhan 1964, 349.
[279] Kittler 1992, 260. Bezeichnenderweise erwähnt Kittler auf der selben Seite McLuhan namentlich.
[280] Vgl. von Baer 1860, 16-26.
[281] So schreibt Kittler in einer etwas schärferen Formulierung seiner bekannten Diagnose: „Jedes Interface unterläuft Wahrnehmungsschwellen und trägt seinen Namen Interface aus purem Spott.“ Vgl. Kittler 1992, 256.

Ein weiterer beachtenswerter Punkt für das Konzept der Sphäre der Berechenbarkeit ist, dass dieser Ansatz das Problem der Repräsentation von »Welt« vorerst ausklammert. Dies hat den Vorteil, dass auf philosophisch problematische, weil noch unzureichend geklärte Annahmen über die Beschaffenheit von »Welt« und Geist, wie sie für Fragen wie der nach dem Verhältnis von klassischem und algorithmischem Zeichen[282] implizit oder explizit immer schon im Voraus getroffen sein müssen, weitestgehend verzichtet werden kann. Ausgehend von dem explizierten Konzept der Sphäre der Berechenbarkeit ist es in umgekehrter Vorgehensweise möglich, die Analyse der Maschine als Ausgangspunkt für eine Bewertung der Adäquatheit der Repräsentation von »Welt« durch (digitale) Maschinen zu verwenden. Die Hoffnung ist, dass dieses umgekehrte Vorgehen die zu klärenden Sachverhalte reduziert, die Problematik einer Betrachtung der Maschine entschärft und die Gefahr minimiert, wieder und wieder in die „Falle philosophischer Selbstermächtigung zu laufen."[283]

Es wurde bereits gezeigt, wie sich die Zuschreibung Gödels, die Turingmaschine sei die Definition all dessen, was mittels einer mechanischen Prozedur ausführbar ist, für eine Analyse der Maschine fruchtbar machen lässt. Interessanterweise verwendet auch Turing einen ähnlichen Begriff. Dieser wirft nicht nur Licht auf die Funktionsweise der Maschine, sondern aus dieser Perspektive auch auf die Frage der Erfassung. Der von Turing dabei verwendete Begriff ist der der *Faustregel*. Er schreibt:

> Die Praxis zeigt, daß LCMs [Logical Computing Machines] alles können, was als >Faustregel< oder als >rein mechanisch< beschreibbar ist. Das ist so ausreichend begründet, daß unter den Logikern nun als ausgemacht gilt, daß >berechenbar mit den Mitteln einer LCM< die genau zutreffende Übersetzung solcher Redewendungen ist.[284]

Dabei ist es keineswegs so, dass nur reale Maschinen nach Faustregeln arbeiten würden. Vielmehr ist dieses Vorgehen im theoretischen Konzept der Turingmaschine verankert:

> Wenn wir die Eigenschaften der Universalmaschine und den Umstand zusammennehmen, daß Maschinenprozesse und strikt nach Faustregeln ablaufende Prozesse gleichbedeutend sind, können wir sagen, daß die Universalmaschine eine Maschine ist, die, wenn sie mit den passenden Befehlen gefüttert wird, zur Ausführung jedes Faustregelprozesses gebracht werden kann.[285]

---

[282] Vgl. Nake 2001 und 2004.

[283] Kittler 1994b, 33.

[284] Turing 1968, 88. Der zitierte Aufsatz Turings mit dem Titel ‚Intelligente Maschinen' erwies sich als Grundlegung der Theorie und der Umsetzung künstlicher Intelligenz *avant la lettre* (vgl. Copeland/Proudfoot 1999). Der Direktor des National Physical Laboratory in London, an dem Turing zu dieser Zeit angestellt war, lehnte diese Arbeit allerdings verächtlich als ein „schoolboy essay" ab (ebd., 99). Dass es sich bei diesem Direktor um Sir Charles Darwin handelte, keinen Geringeren als den Enkel des gleichnamigen englischen Naturforschers, muss vor diesem Hintergrund als Ironie der Geschichte anmuten.

[285] Turing 1947, 193. Weitere Verwendungen des Begriffs Faustregel finden sich unter anderem ebd., 186, 193, 195 und 206 sowie in Turing 1951, 13. Diese Angaben erheben keinen Anspruch auf Vollständigkeit.

Gödels mechanische Prozedur und Turings Faustregelprozeß sind also deckungsgleich. Die Formulierung *Faustregel*, im englischen Original *rule of thumb*, im Deutschen entsprechend auch durch die synonymen Wendungen *Pi mal Daumen* oder *Über den Daumen gepeilt* ausdrückbar,[286] erhält dabei durch folgende Eigenart digitaler Maschinen ihre Berechtigung:

> Es will scheinen, als ob es bei gegebenem Anfangszustand der Maschine und gegebenen Eingabesignalen immer möglich sei, alle zukünftigen Zustände vorherzusagen. Das erinnert an die Laplacesche Ansicht, daß es möglich sein müßte, aus dem vollständigen Zustand des Universums zu einem bestimmten Zeitpunkt, beschrieben durch Lage und Geschwindigkeitkeiten sämtlicher Partikel, alle zukünftigen Zustände vorherzusagen. Die von uns hier betrachtete Vorhersage ist jedoch praktikabler als die von Laplace erwogene. Das System des »Universum als ganzem« ist so beschaffen, daß minimale Fehler in den Anfangsbedingungen zu einem späteren Zeitpunkt einen überwältigenden Einfluß haben können. Die Verschiebung eines einzigen Elektrons um einen billionstel Zentimeter in einem Augenblick könnte ein Jahr später darüber entscheiden, ob ein Mensch von einer Lawine getötet wird oder ihr entkommt. Es ist eine wesentliche Eigenschaft der mechanischen Systeme, die wir »diskrete Maschinen« genannt haben, daß dieses Phänomen nicht auftritt. Selbst wenn wir die tatsächlichen, physikalischen Maschinen anstelle der idealisierten Maschinen betrachten, ergibt sich aus einer verhältnismäßig genauen Kenntnis des jeweiligen Zustandes eine verhältnismäßig genaue Kenntnis aller späteren Schritte.[287]

Der Trick des Digitalen gegenüber dem Analogen ist es ja gerade, zwischen tatsächlich verschiedenen Signalen eine Identität herzustellen, d. h. potentiell unendlich viele singuläre *Ereignisse* zu einem endlichen Zeichenalphabet umzuformen und so als *Serie* nutzbar zu machen.[288] Dies bedeutet, dass Computer *per se* als exakte Maschinen angelegt sind.[289]

---

[286] Der hierbei offen zu Tage liegende Bezug zu den Fingern und zur Hand ist noch zu klären. Gerade vor dem Hintergrund der Wurzel des Begriffs des Digitalen im lateinischen *digitus* sollte die Verbindung zwischen der Formulierung Turings von der *rule of thumb*, ihren deutschen Übersetzungen und der Hand als Rechenhilfe genauer untersucht werden. Die Vermutung liegt nahe, dass der moderne Rechner weniger die Umsetzung eines Denkens ist – zumindest nicht eines Denkens im Sinne Heideggers – als eines *Handhabens* im doppelten Sinne.

[287] Turing 1950, 157f. Auch Kittler zitiert diese Stelle in 1994a, 182f., allerdings mit Abschreibefehlern.

[288] Da Computer aus Sicht der Berechenbarkeit nicht einem speziellen Milieu wie der Elektrizität zuordenbar sind, wurde die Beschreibung hier allgemein gehalten. Im Fall der realen elektronischen Maschinen wird dieser Vorgang umgesetzt, indem die tatsächlich verschiedenen Spannungsverläufe nur zu diskreten und fest getakteten Zeitpunkten gemessen und die sich dabei ergebenden stets unterschiedlichen Messwerte vom Computer als identische Zeichen der einen oder der anderen Gruppe aufgefasst werden, je nachdem, in welches wohldefinierte Intervall von Spannungswerten diese fallen. So könnten beispielsweise Werte zwischen 0 und 1/3 der Versorgungsspannung einer logischen 0 entsprechen, Werte zwischen 2/3 und der vollen Versorgungsspannung einer logischen 1 (so beim CMOS). Der Rechner benötigt so keine exakte Spannung, die ohnehin nicht umzusetzen wäre, sondern stellt diese Exaktheit und damit die Digitalität im Messvorgang selbst her. Dies ist der wesentliche Unterschied zu analogen Geräten, in denen der Signalverlauf stets singuläres Ereignis bleibt und es auf Seiten des sogenannten Menschen (oder eines externen AD-Wandlers) liegt, das entstandene Ergebnis ab- und zurechtzulesen. In diesem Sinne ist auch zu überlegen, ob digitale Rechner auf der grundlegenden Ebene nicht doch in einem operativen Sinne „lesen“, indem sie stets verschiedene tatsächliche Signale (vergleichbar den unter dem Mikroskop stets unterschiedlichen Buchstaben auf einer gedruckten Buchseite) zu Gruppen von identischen Zeichen zusammenziehen (wie es der sogenannte Mensch beim Lesen gewohnt ist) und so erst das Spiel von Identität und Differenz auf den maschinellen Standard bringen.

[289] Natürlich kann ein Computer programmiert werden, die Zeichenfolge ‚ungefähr’ vor jeden ausgegebenen Wert zu setzen. Es ist dabei aber anzunehmen, dass der semantische Gehalt dieses Wortes dem Rechner ebenso verborgen bleiben wird wie ‚ungefähr’ selbst eine exakte Zeichenfolge ist, die vom Computer eben nicht nur ungefähr so ausgegeben wird. Ist der Schritt von der unendlichen Menge verschiedener analoger Signale zum endlichen Alphabet des Digitalen einmal vollzogen, so bleibt streng technisch kein Raum mehr für das Ungefähre – was nicht heißen soll, dass Rechner nicht auf einer höheren Ebene das Ungefähre aufgrund ihrer geradezu wahnwitzigen Rechengeschwindigkeit imitieren und simulieren können.

Was trägt Turings Begriff der Faustregel zur Frage der Erfassung durch digitale Geräten bei? Eine Faustregel anwenden heißt, in Absehung aller möglichen Unschärfen oder der tatsächlichen genauen Lage ein *Schema* heranzuziehen, von dem zumindest erwartet wird, das es hinreichend genaue Ergebnisse liefert.[290] Deswegen aber zu meinen, dieses Arbeiten nach einer Faustregel sei ungenau, heißt, sich grundlegend über den Kern dieses Vorgehens, und damit auch über den Kern der digitalen Maschinen zu täuschen. Ganz im Gegenteil sind Faustregeln *stets absolut exakt*. Darin allein liegt ihre Stärke begründet. Zwar mag das *Ergebnis* der Handhabung von Faustregeln der Lage in der tatsächlichen »Welt« eben nur hinreichend entsprechen – und bei inadäquaten Faustregeln nicht einmal das. Faustregeln als *Methode* aber erhalten ihre Berechtigung eben dadurch, dass sie auch bei mangelnder Kenntnis der genauen Lage in der »Welt« – oder in Fällen, in denen diese Lage zwar bekannt, die Verwendung tatsächlich »welt«-lagegetreuer Methoden aber schlicht ineffizient oder gar ineffektiv wäre – als exakte Methoden zur Verfügung stehen.

Dies mag ungewohnt klingen, aber nur solange, als die Maschine als Verkörperung der Faustregel weiterhin von »Welt« aus gedacht werden soll. Wird die Frage nach der Adäquatheit der Repräsentation von »Welt« durch digitale Maschinen aber eben nicht an den Anfang der Betrachtung gestellt, sondern die Rechner als Phänomen mit eigenem Recht analysiert, so zeigt sich, dass diese nicht anders können, als exakt zu arbeiten. Dafür spricht auch, dass das Netz zwischen »Welt« und digitalen Daten keineswegs so fest ist, wie es immer dann scheinen muss, wenn die Frage nach dem Verhältnis zwischen diesen Elementen von »Welt« und Geist aus gedacht wird. Wenn beispielsweise Stephen Wolfram Komplexitätsexperimente mit zellulären Automaten vollzieht,[291] dann hat diese Phänomenotechnik im Sinne Bachelards[292] nicht länger einen im klassischen Sinne repräsentationalen Bezug zur »Welt«. Hier steht kein Zeichen mehr repräsentativ für eine naiv-reale Sache außerhalb des Rechners.

Ob man dabei allerdings nicht nur *methodisch*, sondern auch *ontologisch* so weit gehen kann, die Maschinen primär als präzedierende Simulakren zu begreifen, und erst sekundär als Modellierung und Repräsentation von »Welt«, ob also Baudrillards Satz: „Die Karte ist dem Territorium vorgelagert, ja sie bringt es erst hervor“[293] auch hier Geltung verlangen darf, soll in dieser Arbeit dahingestellt bleiben.

---

[290] Daher auch die Formulierung vom *Schema F*, die ebenfalls zu Faustregel synonym ist.
[291] Vgl. Wolfram 2002 und Martínez 2004.
[292] Vgl. Castelao-Lawless 1995, besonders 51.
[293] Baudrillard 1978, 8.

7. Diesseits und Jenseits der mechanischen Prozedur

Klar geworden ist, dass die Sphäre der Berechenbarkeit all das umfasst, was digitale Medien leisten können. In diesem Sinne ist das, was von der Turingmaschine definiert wird, d. h. all das, was mittels einer mechanischen Prozedur bzw. in einem Faustregelprozess durchgeführt werden kann, äquivalent zu allem, für das sich überhaupt ein Schema finden lässt, auf dessen Grundlage es sich *operativ* in Gang setzen lässt, ungeachtet dessen, ob es tatsächlich im Realen abgeschlossen werden kann.

Die Turingmaschine als Definition des operativ Umsetzbaren.

An diesen „Spielregeln" des Digitalen, wie sie diese Arbeit beleuchtete, wird sich, „[…] solange Turingmaschinen, Digitalcomputer und Millennium Bugs insistieren, als sei alle Technikgeschichte am theoretisch-praktischen Ende angelangt, […] wenig ändern."[294] Die Frage ist daher, ob nicht auch die Modelle der Hypercomputation am Ende nur ins Unendliche gesteigerte mechanische Prozeduren sind. Vielleicht muss die Suche nach systematischen Prozeduren, die eben nicht mechanischer Natur sind, an ganz anderen Stellen ansetzen. So oder so wird sich zeigen, ob Kittler mit seiner Diagnose letztendlich richtig lag, wenn er schrieb: „Alle Technologien haben sich als verwundbar, als überbietbar erwiesen – warum sollten Computer die einzige Ausnahme machen?"[295]

[294] Kittler 2000, 267.
[295] Ebd., 268.

8. Literaturverzeichnis

Adamatzky, Andrew (Hg.) (2010): ***Game of Life Cellular Automata***. London: Springer.

Aristoteles: ***Physikvorlesung***. Werke in deutscher Übersetzung Band 11. Dritte, unveränderte, durch eine Nachbemerkung ergänzte Auflage 1979. Übersetzt von Hans Wagner. Herausgegeben von Hans Grumach und Hellmut Flashar. Darmstadt: Akademie-Verlag.

Baer, Karl Ernst von (1860): **‚Welche Auffassung der lebenden Natur ist die richtige? und Wie ist diese Auffassung auf die Entomologie anzuwenden?'** Zur Eröffnung der Russischen entomologischen Gesellschaft, im Mai 1860, gesprochen von Karl Ernst v. Baer, ihrem derzeitigen Präsidenten.

Bakker, C./Comberg, W./Dold, H./Frey, E./Igersheimer, J./Kümmell, R./Lenz, G./Lichtwitz, L./Lutz, W./Sattler, C. H./Schieck, F./Steidle, H./Zade, M./Zondek, M. (1932): ***Auge und Allgemeinleiden. Therapie; Hygiene***. 7. Band. Berlin: Springer.

Baudrillard, Jean (1978): ***Agonie des Realen***. Berlin: Merve.

Benacerraf, Paul (1962): **‚Tasks, Super-Tasks and the Modern Eleatics'**. In: *The Journal of Philosophy*, Vol. 59, No. 24, 765-784.

Berlin-Brandenburgische Akademie der Wissenschaften (Hg.) (2016): ***Jahrbuch 2015***. Berlin: Berlin-Brandenburgische Akademie der Wissenschaften.

Blake, R. M. (1926): **‚The Paradox of Temporal Process'**. In: *Journal of Philosophy* 23, 645-654.

Bolz, Norbert/Kittler, Friedrich A./Tholen, Christoph (Hg.) (1999): ***Computer als Medium***. *Literatur- und Medienanalysen Band 4*. München: Wilhelm Fink.

Boole, George (1847): ***The Mathematical Analysis of Logic***. Reprint 1998. Bristol/Sterling: Thoemmes.

Braun, Hans-Joachim/Kaiser, Walter (1990): ***Energiewirtschaft. Automatisierung. Information. Seit 1914***. Propyläen Technikgeschichte. Herausgegeben von Wolfgang König. Fünfter Band. Unveränderte Neuausgabe 1997. Berlin: Propyläen (Ullstein).

Calude, Cristian S./Pavlov, Boris (2002): ***Coins, Quantum Measurements and Turing's Barrier***.
Download am 9.9.16 von: https://arxiv.org/pdf/quant-ph/0112087v3.pdf

Castelao-Lawless, Teresa (1995): **‚Phenomenotechnique in Historical Perspective: Its Origins and Implications for Philosophy of Science'**. In: *Philosophy of Science*, Vol. 62, No. 1 (March 1995), 44-59.
Download am 27.11.15 von: http://www.jstor.org/stable/188034

Chaitin, Gregory J. (1975): **‚A Theory of Program Size Formally Identical to Information Theory'**. In: *Journal of the Association for Computing Machinery*, Vol. 22, No. 3, Juli 1975, 329-340.

Chaitin, Gregory J. (1990): **‚Lecture – Undecidability & Randomness in Pure Mathematics'**. In: Ders. (2002): *Conversations with a Mathematician. Math, Art, Science and the Limits of Reason*. London u. a.: Springer 2002, 113-126.

Chaitin, Gregory J. (2000): **‚Lecture – A Century of Controvery over the Foundations of Mathematics'**. In: Ders. (2002): *Conversations with a Mathematician. Math, Art, Science and the Limits of Reason*. London u. a.: Springer 2002, 5-40.

Copeland, B. Jack (2002): **‚Accelerating Turing Machines'**. In: *Minds and Machines* 12, 281-301.

Copeland, B. Jack (Hg.) (2004): ***The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus The Secrets of Enigma***. Oxford/New York: Oxford University Press.

Copeland, B. Jack/Proudfoot, Diane (1999): **‚Alan Turing's Forgotten Ideas in Computer Science'**. In: *Scientific American*, 253:4, 98-103.

Copeland, B. Jack/Shagrir, Oron (2015): **‚Turing versus Gödel on Computability and the Mind'**. In: Copeland, B. Jack/Posy, Carl J./Shagrir, Oron (Hg.): *Computability. Turing, Gödel, Church and Beyond*. Cambridge/London: MIT Press, 1-33.

Copeland, B. Jack/Shagrir, Oron (2007): **‚Physical Computation: How General are Gandy's Principles for Mechanisms'**. In: *Minds & Machines*, 17, 217-231.

Davies, E. B. (2001): **‚Building Infinite Machines'**. In: *The British Journal for the Philosophy of Science*, 52 (4), 671-682.
Download (mit divergierender Seitennummerierung, hier: 1-11) am 13.8.15 von: http://www.mth.kcl.ac.uk/staff/eb_davies/jphilsci.pdf

Davis, Martin (1958): ***Computability and Unsolvability***. New York/Toronto/London: McGraw-Hill.

Deutsch, D. (1985): **‚Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer'**. In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Vol. 400, No. 1818 (Jul. 8, 1985), 97-117.

Dijksterhuis, E. J. (1951): **‚Die Mechanisierung des Weltbildes'**. Vortrag vom 26. November 1951 vor der Philosophischen Gesellschaft und dem Physikalischen Verein in Bremen. Gekürzt und neu abgedruckt in: *Physikalische Blätter*, 12. Jahrgang 1956/Heft 11, 481-494.

Dotzler, Bernhard/Kittler, Friedrich (1987): ***Alan M. Turing. Intelligence Service. Schriften***. Berlin: Brinkmann & Bose.

Earman, John/Norton, John D. (1996): **‚Infinite Pains: The Trouble with Supertasks'**. In: Morton, Adam/Stich, Stephen P. (Hg.) (1996): *Benacerraf and his Critics*. Oxford: Blackwell, 231-261.

Ernst, Wolfgang (2009a): **‚Die Frage nach dem Zeitkritischen'**. In: Volmar, Axel (2009): *Zeitkritische Medien*. Berliner Programm einer Medienwissenschaft. Band 5. Berlin: Kulturverlag Kadmos, 27-42.

Ernst, Wolfgang (2009b): ***„Else loop forever". The Untimeliness of Media***. Vortragstext, o. S.
Download am 11.7.15 von: https://www.medienwissenschaft.hu-berlin.de/de/medienwissenschaft/medientheorien/downloads/publikationen/ernst-else-loop-forever.pdf

Ernst, Wolfgang (2012a): ***Chronopoetik. Zeitweisen und Zeitgaben technischer Medien***. Berliner Programm einer Medienwissenschaft. Band 10. Berlin: Kulturverlag Kadmos.

Ernst, Wolfgang (2012b): ***Gleichursprünglichkeit. Zeitwesen und Zeitgegebenheit technischer Medien***. Berliner Programm einer Medienwissenschaft. Band 11. Berlin: Kulturverlag Kadmos.

Ernst, Wolfgang (2012c): **‚Logos der Maschine'**. In: Herrmann, Hans-Christian von/Velminski, Wladimir (Hg.): *Maschinentheorien/Theoriemaschinen*. Frankfurt a. M. et al.: Peter Lang.

Ernst, Wolfgang/Maibaum, Johannes/Völz, Horst (Hg.) (2016 [voraussichtliches Erscheinen]): ***Horst Völz: Speichertheorie. Ausgewählte Schriften zur Elektronik, Informatik und Kultur technischer Speicher***. Berliner Programm einer Medienwissenschaft Band 13. Berlin: Kulturverlag Kadmos.

Etesi, Gábor/Németi, István (2002): ***Non-Turing computations via Malament-Hogarth space-times***.
Download am 9.9.16 von https://arxiv.org/pdf/gr-qc/0104023.pdf

Farhi, Edward/Goldstone, Jeffrey (2000): ***Quantum Computation by Adiabatic Evolution***.
Download am 25.7.16 von: http://arxiv.org/pdf/quant-ph/0001106.pdf

Feynman, Richard P. (1982): **‚Simulating Physics with Computers'**. In: *International Journal of Theoretical Physics*, Vol. 21, Nos. 6/7, 467-488.

Flammarion, Nicolas Camille (1888): ***L'atmosphère. Météorologie populaire***. Paris: Hachette.

Frey, Peter (2009): ***Kommunizierende Automaten. Die Dynamisierung der Schrift als medientheoretische Zäsur***. Bielefeld: Transcript.

Gandy, Robin (1980): **‚Church's Thesis and Principles for Mechanisms'**. In: Barwise, J./Keisler, H.J./Kunen, K. (Hg.): *Studies in Logic and the Foundations of Mathematics. Volume 101. The Kleene Symposium*. Amsterdam: North-Holland, 123-148.

Galilei, Galileo (1613): **‚Brief an D. Benedetto Castelli (21. Dezember 1613)'**. Wiederabdruck in: Mudry, Anna (Hg.) (1987): *Galileo Galilei. Schriften – Briefe. Dokumente*. Band 1. Schriften. Berlin: Rütten & Loening, 168-177.

Gödel, Kurt (1931): **‚Über formal unentscheidbare Sätze der *Principia mathematica* und verwandter Systeme I'**. Wiederabdruck in: Feferman, Solomon, et al. (Hg.) (1986): *Kurt Gödel. Collected Works. Volume I. Publications 1929-1936*. New York/Oxford: Oxford University Press/Clarendon Press, 144-194.

Gödel, Kurt (193?): **‚Undecidable diophantine propositions'**. Wiederabdruck in: Feferman, Solomon, et al. (Hg.) (1995): *Kurt Gödel. Collected Works. Volume III. Unpublished essays and lectures*. New York/Oxford: Oxford University Press, 164-175.

Gödel, Kurt (1951): **‚Some basic theorems on the foundations of mathematics and their implications'**. Wiederabdruck in: Feferman, Solomon, et al. (Hg.) (1995): *Kurt Gödel. Collected Works. Volume III. Unpublished essays and lectures*. New York/Oxford: Oxford University Press, 304-323.

Gödel, Kurt (1964): **‚Postscriptum to Gödel 1934'**. Wiederabdruck in: Feferman, Solomon, et al. (Hg.) (1986): *Kurt Gödel. Collected Works. Volume I. Publications 1929-1936*. New York/Oxford: Oxford University Press/Clarendon Press, 369-371.

Gödel, Kurt (1972): **‚On an extension of finitary mathematics which has not yet been used'**. Wiederabdruck in: Feferman, Solomon, et al. (Hg.) (1990): *Kurt Gödel. Collected Works. Volume II. Publications 1938-1974*. New York/Oxford: Oxford University Press/Clarendon Press, 271-280.

Hagar, A./Korolev, A. (2007): **‚Quantum hypercomputation: Hype or Computation?'**. In: *Philosophy of Science*, 74 (3), 347-363.

Hagen, Wolfgang (2002): **‚Die Camouflage der Kybernetik'**. Wiederabdruck in: Pias, Claus (Hg.) (2004): *Cybernetics – Kybernetik. The Macy-Conferences 1946-1953 – Die Macy-Konferenzen 1946-1953. Vol. II – Band II. Essays & Documents / Essays & Dokumente*. Zürich/Berlin: Diaphanes, 191-207.

Heidegger, Martin (1927): ***Sein und Zeit***. Neunzehnte Auflage 2006. Tübingen: Max Niemeyer.

Heidegger, Martin (1962): ***Die Technik und die Kehre***. Zwölfte Auflage 2011. Stuttgart: Klett-Cotta.

Heintz, Bettina (1993): ***Die Herrschaft der Regel***. Frankfurt a. M./New York: Campus.

Hodges, Andrew (1983): ***Alan Turing, Enigma***. Vollständige Übersetzung, 1989. Übersetzt von Rolf Herken und Eva Lack. Berlin: Kammerer & Unverzagt.

Hodges, Andrew (2006): ***Can quantum computing solve classically unsolvable problems?*** Download am 25.7.16 von: http://arxiv.org/pdf/quant-ph/0512248.pdf

Hofstadter, Douglas R. (1979): ***Gödel Escher Bach ein Endloses Geflochtenes Band***. Aus dem Englischen übersetzt. 12. Auflage September 2008. München: Klett-Cotta.

Hogarth, Mark L. (1992): **‚Does General Relativity Allow an Observer to View an Eternity in a Finite Time?'**. In: *Foundations of Physics Letters*, 5, 173-181.

Hogarth, Mark L. (1994): **‚Non-Turing Computers and Non-Turing Computability'**. In: *Philosophy of Science Association*, 1, 126-138.

Khan, Douglas (2013): ***Earth Sound Earth Signal. Energies and Earth Magnitude in the Arts***. Berkeley/Los Angeles/London: University of California Press.
Kieu, T. D. (2003a): ***A reformulation of Hilbert's tenth problem through Quantum Mechanics***.
Download am 25.7.16 von: http://arxiv.org/pdf/quant-ph/0111063.pdf

Kieu, T. D. (2003b): ***Quantum Algorithm for Hilbert's Tenth Problem***.
Download am 25.7.16 von: https://arxiv.org/pdf/quant-ph/0110136.pdf

Kieu, T. D. (2003c): ***Computing the Noncomputable***.
Download am 25.7.16 von: http://arxiv.org/pdf/quant-ph/0203034.pdf

Kieu, T. D. (2003d): ***Quantum Adiabatic Algorithm for Hilbert's Tenth Problem: I. The Algorithm***.
Download am 25.7.16 von: http://arxiv.org/pdf/quant-ph/0310052.pdf

Kieu, T. D. (2004): ***An Anatomy of a Quantum Adiabatic Algorithm that Transcends the Turing Computability***.
Download am 25.7.16 von: http://arxiv.org/pdf/quant-ph/0407090.pdf

Kittler, Friedrich A. (1986): ***Grammophon – Film – Typewriter***. Berlin: Brinkmann & Bose.

Kittler, Friedrich A. (1988): **‚Die Stadt ist ein Medium'**. Wiederabdruck in: Ders. (2013): *Die Wahrheit der technischen Welt. Essays zur Genealogie der Gegenwart*. 2. Auflage 2014. Herausgegeben und mit einem Nachwort von Hans Ulrich Gumbrecht. Berlin: Suhrkamp, 181-197.

Kittler, Friedrich A. (1990): **‚Die künstliche Intelligenz des Weltkriegs: Alan Turing'**. Wiederabdruck in: Ders. (2013): *Die Wahrheit der technischen Welt. Essays zur Genealogie der Gegenwart*. 2. Auflage 2014. Herausgegeben und mit einem Nachwort von Hans Ulrich Gumbrecht. Berlin: Suhrkamp, 232-252.

Kittler, Friedrich A. (1992): **‚Gleichschaltungen. Über Normen und Standards der elektronischen Kommunikation'**. Wiederabdruck in: Faßler, Manfred/Halbach, Wulf R. (Hg.) (1998): *Geschichte der Medien*. München: Wilhelm Fink, 255-267.

Kittler, Friedrich A. (1993) [1992 erschienen als ‚There is no software']: **‚Es gibt keine Software'**. Wiederabdruck in: Ders. (2013): *Die Wahrheit der technischen Welt. Essays zur Genealogie der Gegenwart*. 2. Auflage 2014. Herausgegeben und mit einem Nachwort von Hans Ulrich Gumbrecht. Berlin: Suhrkamp, 285-299.

Kittler, Friedrich A. (1994a): **‚Die Zukunft auf Siliziumbasis'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 177-197.

Kittler, Friedrich A. (1994b): **‚Spiele des Wahren und Falschen. Zum zehnten Todestag des französischen Philosophen Michel Foucault'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 31-40.

Kittler, Friedrich A. (1995a): **‚Der Kopf schrumpft. Herren und Knechte im Cyberspace'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 168-176.
Kittler, Friedrich A. (1995b): **‚Die Informationsbombe. Gespräch mit Paul Virilio'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 134-161.

Kittler, Friedrich A. (1995c): **‚Museen an der digitalen Grenze'.** In: Helas, Philine et. al. (Hg.) (2007): *Bild/Geschichte. Festschrift für Horst Bredekamp*. Berlin: Akademie, 109-118.

Kittler, Friedrich A. (1996a): **‚Computeranalphabetismus'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 109-133.

Kittler, Friedrich A. (1996b): **‚Provisorische Maschinen, provisorische Moral. Philosophie unter maschinellem Konkurrenzdruck'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 162-167.

Kittler, Friedrich A. (1997): **‚Memories are made of you'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 41-67.

Kittler, Friedrich A. (2000): **‚Krieg im Schaltkreis. Die Herren der Software sind die Condottiere der neuen Kampfzonen: Die Spielkonsole fliegt den Kampfjet'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 249-268.

Kittler, Friedrich A. (2002): **‚Das Ganze steuert der Blitz. Gespräch mit Alexander Kluge'**. In: Ders. (2002): *Short Cuts*. Herausgegeben von Peter Gente und Martin Weinmann. Frankfurt a. M.: Zweitausendeins, 269-282.

Kuck, David J. (1989): **‚The von Neumann and Parallel Computing Eras'**. In: Ritter, G. X. (Hg.) (1989): *Information Processing 89. Proceedings of the IFIP 11th World Computer Congress. San Francisco, U.S.A. August 28–September 1, 1989*. Amsterdam/New York/Oxford/Tokyo: North Holland, 230.

Lacan, Jacques (1955): **‚Psychoanalyse und Kybernetik oder von der Natur der Sprache'**. Vortrag vom 22. Juni 1955. Wiederabdruck in: Ders. (1991): *Das Seminar von Jacques Lacan. Buch II (1954-1955). Das Ich in der Theorie Freuds und in der Technik der Psychoanalyse*. Übersetzt von Hans-Joachim Metzger. Weinheim/Berlin: Quadriga, 373-390.

Lagrange, Joseph-Louis (1788): ***Analytische Mechanik***. Deutsch herausgegeben von Dr. H. Servus, 1887. Berlin: Springer.

Lichtenberg, Georg Christoph [Verfasst 1765-1799]: ***Aphorismen***. Nach der Ausgabe seiner Söhne neu zusammengestellt und herausgegeben von Klaus Bock, 1990. Kettwig: Phaidon.

Margolus, Norman/Toffoli, Tommaso (1987a): **‚Cellular Automata Machines'**. In: *Complex Systems* 1 (1987), 967-993.

Margolus, Norman/Toffoli, Tommaso (1987b): ***Cellular Automata Machines. A new environment for modelling***. Cambridge, London: MIT Press.

Martínez, Genaro Juárez (2004): ***Introduction to Rule 110***.
Download am 13.10.15 von:
http://www.rule110.org/amhso/results/rule110-intro/introRule110.html

Mayer-Lindenberg, Fritz (1998): ***Konstruktion digitaler Systeme. Eine kurze Einführung in die Informatik***. Braunschweig: Vieweg.

McLuhan, Marshall (1964): ***Understanding Media. The Extensions of Man***. London/New York: Ark Paperbacks.

McLuhan, Marshall/Fiore, Quentin (1967): ***The Medium is the Massage***. Renewed 1996 by Jerome Agel. London: Penguin.

McLuhan, Marshall/Fiore, Quentin (1968): ***War and Peace in the Global Village***. Renewed 1997 by Jerome Agel. Auflage von 2001. Corte Madera: Gingko Press.

Mill, John Stuart (1843): ***A system of logic. Ratiocinative and Inductive***. Reprint 1996. Toronto: Routledge.

Nagel, Ernest/Newman, James R. (1958): ***Der Gödelsche Beweis***. 9., unveränderte Auflage 2010. München: Oldenbourg.

Nake, Frieder (2001): **‚Das algorithmische Zeichen'**. In: Bauknecht, W./Brauer, W./Mück, Th. (Hg.): *Informatik 2001. Tagungsband der GI/OCG 2001*. Band II. Wien: Österreichische Computergesellschaft, 736-742.

Nake, Frieder (2004): **‚Das algorithmische Zeichen und die Maschine'**. In: Paul, Hansjürgen/Latniak, Erich (Hg.): *Perspektiven der Gestaltung von Arbeit und Technik. Festschrift für Peter Brödner*. Arbeit und Technik. Schriftenreihe des Instituts Arbeit und Technik im Wissenschaftszentrum Nordrhein-Westfalen. München/Mering: Rainer Hampp, 203-223.

Neumann, Johann von (1927): **‚Zur Hilbertschen Beweistheorie'**. In: *Mathematische Zeitschrift* 26, 1-46.

Ord, Toby (2002): ***Hypercomputation: computing more than the Turing machine***.
Download am 8.9.16 von: http://arxiv.org/pdf/math/0209332.pdf

Pias, Claus (2015): **‚Friedrich Kittler und der »Mißbrauch von Heeresgerät«. Zur Situation eines Denkbildes 1964 – 1984 – 2014'**. In: *Merkur. Deutsche Zeitschrift für europäisches Denken*. 69. Jahrgang, April 2015, 31-44.

Platon: **‚Alkibiades I'**. In: Wolf, Ursula (Hg.) (2007): *Platon. Sämtliche Werke*. Band 1. Auf Grundlage der deutschen Übersetzung von Friedrich Schleiermacher. 30. Auflage. Reinbek bei Hamburg: Rowohlt, 123-180.

Post, Emil L. (1936): **‚Finite Combinatory Processes-Formulation 1'**. In: *The Journal of Symbolic Logic*, Vol. 1, No. 3. (September 1936), 103-105.

Priestley, Mark (2011): ***A Science of Operations. Machines, Logic and the Invention of Programming***. London: Springer.

Rendell, Paul (2001): ***A Turing Machine In Conway's Game Life***.
Download am 3.9.16 von:
https://www.ics.uci.edu/~welling/teaching/271fall09/Turing-Machine-Life.pdf

Ritter, G. X. (Hg.) (1989): ***Information Processing 89. Proceedings of the IFIP 11th World Computer Congress. San Francisco, U.S.A. August 28–September 1, 1989***.
Amsterdam/New York/Oxford/Tokyo: North Holland.

Russell, Bertrand (1915): ***Our Knowledge of the External World as a Field for Scientific Method in Philosophy***. London: George Allen & Unwin.

Russell, Bertrand (1936): **‚The Limits of Empiricism'**. In: *Proceedings of the Aristotelian Society* 36, 131-150.

Shagrir, Oron (2003): **‚Super-tasks, accelerating Turing machines and uncomputability'**.
In: *Theoretical Computer Science*, 317, 105-114.

Shagrir, Oron (2006): **‚Gödel on Turing on Computability'**. In: Olszewski, Adam/Wolenski, Jan/Janusz, Robert (Hg.): *Church's Thesis after 70 Years*.
Frankfurt/Paris/Ebikon/Lancaster/New Brunswick: Ontos, 393-419.

Shannon, Claude Elwood (1938): **‚A Symbolic Analysis of Relay and Switching Circuits'**.
In: *Transactions of the American Institute of Electrical Engineers* 57 (1938), 471-495.

Sieg, Wilfried (2006): **‚Gödel on computability**'. In: *Philosophia Mathematica* 14 (2): 189-207.
Download (mit divergierender Seitennummerierung, hier: 1-21) am 11.7.15 von:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.365&rep=rep1&type=pdf

Smith, Warren D. (2006): **‚Three counterexamples refuting Kieu's plan for ''quantum adiabatic hypercomputation''; and some uncomputable quantum mechanical tasks'**. In: *Applied Mathematics and Computation*, 178, 184-193.

Stewart, Ian (1991): **‚Deciding the Undecidable'**. In: *Nature*, Vol. 352 (22. August 1991), 664-665.

Thomson, James F. (1954): **‚Tasks and Super-Tasks'**. In: *Analysis* 15: 1, 1-13.

Thomson, James F. (1970): **‚Comments on Professor Benacerraf's paper'**. In: Salmon, Wesley C. (Hg.): *Zeno's Paradoxes*. Indianapolis: Bobbs-Merrill, 130-138.

Turing, Alan Mathison (1935): **‚Equivalence of Left and Right Almost Periodicity'**. In: *Journal of the London Mathematical Society*, 10 (1935), 284-285.

Turing, Alan Mathison (1936): **‚On Computable Numbers with an Application to the Entscheidungsproblem'**. Wiederabdruck in: Copeland, B. Jack (Hg.) (2004): *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus The Secrets of Enigma*. Oxford/New York: Oxford University Press, 58-90.

Turing, Alan Mathison (1939): **‚Systems of Logic Based on Ordinals'**. Dissertation von 1938. In: *Proceedings of the London Mathematical Society* (1939) s2-45 (1), 161-228.

Turing, Alan Mathison (1947): **‚Lecture to the London Mathematical Society'**. Gehalten im Februar 1947. Aus dem Englischen übersetzt unter dem Titel ‚The State of the Art' in: Dotzler, Bernhard/Kittler, Friedrich A. (1987): *Alan M. Turing. Intelligence Service. Schriften*. Berlin: Brinkmann & Bose, 183-207.

Turing, Alan Mathison (1950): **‚Computing Machinery and Intelligence'**. Aus dem Englischen übersetzt in: Dotzler, Bernhard/Kittler, Friedrich A. (1987): *Alan M. Turing. Intelligence Service. Schriften*. Berlin: Brinkmann & Bose, 148-182.

Turing, Alan Mathison (1951): **‚Intelligent Machinery, a Heretical Theory'**. Aus dem Englischen übersetzt in: Dotzler, Bernhard/Kittler, Friedrich A. (1987): *Alan M. Turing. Intelligence Service. Schriften*. Berlin: Brinkmann & Bose, 6-15.

Turing, Alan Mathison (1954): **‚Solvable and Unsolvable Problems'**. Wiederabdruck in: Copeland, B. Jack (Hg.) (2004): *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus The Secrets of Enigma*. Oxford/New York: Oxford University Press, 576-595.

Turing, Alan Mathison (1968) [Verfasst 1948]: **‚ Intelligent Machinery'**. Aus dem Englischen übersetzt in: Dotzler, Bernhard/Kittler, Friedrich A. (1987): *Alan M. Turing. Intelligence Service. Schriften*. Berlin: Brinkmann & Bose, 81-113.

Wang, Hao (1987): ***Reflections on Kurt Gödel***. Fourth Printing 1995. Cambridge/London: MIT Press.

Webb, Judson C. (1990): **‚Introductory Note to Remark 3 of Gödel 1972a'**. In: Feferman, Solomon, et al. (Hg.) (1990): *Kurt Gödel. Collected Works. Volume II. Publications 1938-1974*. New York/Oxford: Oxford University Press/Clarendon Press, 292-304.

Weyl, Hermann (1927): ***Philosophie der Mathematik und Naturwissenschaft***. 8. Auflage 2009. Oldenbourg: Scientia Nova.

Weizenbaum, Joseph (1977): ***Die Macht der Computer und die Ohnmacht der Vernunft***. Frankfurt a. M.: Suhrkamp.

Winthrop-Young, Geoffrey (2005): ***Friedrich Kittler zur Einführung***. Hamburg: Junius.

Wischik, Lucian (1997): ***A Formalisation of Non-Finite Computation***. Dissertation an der University of Cambridge.
Download am 9.9.16 von: http://www.wischik.com/lu/philosophy/non-finite-computation.pdf

Wolfram, Stephen (2002): ***A New Kind of Science***. Champaign: Wolfram Media.

Yao, Andrew Chi-Chih (2003): **‚Classical Physics and the Curch-Turing Thesis'**. In: *Journal of the ACM* 50, 100-105.
Download (mit divergierender Seitennummerierung, hier: 1-7) am 12.7.16 von: http://eccc.hpi-web.de/report/2002/062/

Zuse, Konrad (1962): **‚Entwicklungslinien einer Rechengeräte-Entwicklung von der Mechanik zur Elektronik'**. In: Hoffmann, Walter (Hg.) (1962): *Digitale Informationswandler. Probleme der Informationsverarbeitung in ausgewählten Beiträgen*. Braunschweig: Vieweg, 508-532.

9. Internetquellen

Becker, Klaus (2013): **‚Präzisierung der Turingmaschine'**.
Besuch am 21.9.16:
http://www.inf-schule.de/grenzen/berechenbarkeit/turingmaschine/station_turingmaschine

Benvenuto, Santiago (2015): **‚10 impactantes montajes de Mise en Abyme'**.
Besuch am 26.9.16:
http://www.batanga.com/curiosidades/7388/10-impactantes-montajes-de-mise-en-abyme

Bianic, F. (2013): **‚"La fée électricité" 1937 Raoul DUFY.'**
Besuch am 20.9.16:
http://www.collegebrossolette.com/spip.php?article618

Flynn, Laurie J. (2004): **‚Intel Halts Development of 2 New Microprocessors'**.
Besuch am 3.9.2016:
http://www.nytimes.com/2004/05/08/business/intel-halts-development-of-2-new-microprocessors.html

Gauss, Carl (1831): **‚Brief Nr. 396 von Gauss an Schumacher am 12.7.1831 mit einer Bewertung zum Unendlichen in der Mathematik'**. Für das Internet aufbereitet von Rudolf Sponsel.
Besuch am 15.10.16:
http://www.sgipt.org/wisms/geswis/mathe/gsb396.htm

Guizzo, Erico (2010): **‚DIY Turing Machine'**.
Besuch am 20.5.16:
http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/032610-diy-turing-machine

Hagar, Amit/Cuffaro, Michael (2015): **‚Stanford Encyclopedia of Philosophy**: **Quantum Computing'**.
Besuch am 10.9.16:
http://plato.stanford.edu/entries/qt-quantcomp/

Hodges, Andrew: **‚Copeland and Proudfoot's article in Scientific American, April 1999: comment by Andrew Hodges'**.
Besuch am 4.6.16:
http://www.turing.org.uk/publications/sciam.html

‚In vollen Zügen' (2011): **‚Entwicklung von Bildschirmschonern – Part 1: Conway's Game of Life'**.
Besuch am 3.9.16:
https://invollenzuegen.wordpress.com/2011/12/08/entwicklung-von-bildschirmschonern-part-1-conways-game-of-life/

Klug, Heinz G. (2010): **‚Wo das Gehirn versagt. „Nachdenken über Quantenphysik"'**.
Besuch am 11.8.16:
http://www.hg-klug.de/mrganz/versag/versag.html

Manchak, John/Roberts, Bryan W. (2016): **‚Stanford Encyclopedia of Philosophy**: **Supertasks'**.
Besuch am 12.8.16:
http://plato.stanford.edu/entries/spacetime-supertasks/

Nückel, Thomas (2016): **‚Cellular Sounds Project'**.
Besuch am 3.9.16:
https://www.youtube.com/channel/UCAYWbTbWZ5VimoDLYA1CsDg